DEPARTMENT OF THE ARMY
US ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VIRGINIA 22333-5600

(2)

REPLY TO
ATTENTION OF

PERI-BR

MEMORANDUM FOR Marketing and Publication

SUBJECT: Certification and Transmittal of Final Manuscript

1. The enclosed final manuscript is submitted.

   a. Title: Problem Solving and Learning in a Natural Task Domain

   b. First author: Janet Kolodner

   c. Contributing author(s): Lawrence Barsalou

   d. Field unit/tech area: Office of Basic Research

   e. Present FU/TA/HQ office chief: Dr. Michael Kaplan

   f. Project name:

2. Checklist has been completed.

3. It is to be published as a: Rsch Note

4. The DoD Distribution statement is:

5. First-time-distribution list requested is:
   Additional author copies requested:_____.

[illegible]
MICHAEL KAPLAN
Director, Basic Research

6 Encls
1. Package checklist
2. Peer review - 1
3. Report documentation page (DD Form 1473)
4. Table of contents
5. Body of the manuscript
6. Reference list

PEER REVIEW
(Subject matter expert review)

Date Due: ____________
Date Received: 10/30/89
Date Returned: ____________

Reviewer (full name) Dr. Judith Orasanu

Organization Office of Basic Research

Manuscript Title Problem Solving and Learning in a Natural Task Domain

Author or COR Janet Kolodner

I. RATINGS (extent to which criteria are met):

A. FOREWORD, EXECUTIVE SUMMARY (BRIEF): Are they clear? Are they consistent with the contents? Are they directed to target readers or users? Do they concisely highlight the important findings?

No __ Moderately __ [✓ Yes] Substantially __

B. INTRODUCTION, BACKGROUND, OBJECTIVES: Is the literature review relevant to the research conducted (necessary and sufficient)? Is the statement of the problem pertinent to the target audience? Is it clear? Is it supported by concrete data or evidence?

No __ Moderately __ Yes __ [✓ Substantially]

C. APPROACH, METHOD: Is it appropriate? Was there a valid experimental design and data collection plan? Are they competently described and were they competently executed?

No __ Moderately __ Yes __ [✓ Substantially]

D. RESULTS: Are they clearly presented? Is there appropriate use of tables and figures? Were proper statistics used? Were they used correctly?

No __ Moderately __ Yes __ [✓ Substantially]

E. DISCUSSION AND CONCLUSIONS: Do they follow from the data and literature review? Are they comprehensible? Are conclusions and recommendations warranted and usable by intended audience?

No __ Moderately __ Yes __ [✓ Substantially]

ARI Form 185, 20 Oct 88

Peer Review (cont)

II. Recommendations

___ Should not be published.

___ Return for reconsideration (e.g., reanalysis, additional data collection, or rewrite). Should not be published as is. (Comments may be made in Section III or as an enclosure.)

___ Publish after minor revisions. (Comments may be made in Section III or as an enclosure.)

✓ Publish as is.

My name may / may not appear on the inside cover as reviewer.

[illegible]
Reviewer's Signature

III. Comments on any of the above ratings or recommendations.

Accesion For	
NTIS CRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

PUBLICATION CHECKLIST

Use for: Research Report, Technical Report, Research Product, Research Note, Special Report, or book published by ARI.

Type of document (circle one): RR TR RP [✓ RN] SP book

Submit an original and one copy of:

Documentation required for publication

___ 1. Certification DF, signed.

Peer review - 1
___ 2a. Recommended changes made.
___ 2b. Changes not made--reviewer's name not to be used.

Peer review - 2 (only one required for Research Note)
___ 3a. Recommended changes made.
___ 3b. Changes not made--reviewer's name not to be used.

Letter(s) of permission to quote copyrighted material
___ 4a. Required and submitted.
✓ 4b. Not required.

Sensitivity review
___ 5a. Required and submitted.
✓ 5b. Not required.

Security review
___ 6a. Required and submitted.
✓ 6b. Not required.

✓ 7. DD Form 1473, completed.

Manuscript

Foreword
___ 8a. 6.3 research.
___ 8b. Other research funding.
✓ 8c. Not required (Research Note).

Acknowledgment
___ 9a. Included.
✓ 9b. Not included.

Executive summary
___ 10a. Included.
✓ 10b. Not required (RN, RP, or book).

___ 11. Table of contents (with Lists of Tables and Figures).

ARI Form 188, 20 Oct 88

Publication Checklist (cont)

Body of the manuscript

✓ 12. Text of report (with Tables and Figures).
  a. All pages are present and numbered properly.
  b. Text is print-ready copy.
  c. Table of contents accurately indicates configuration of document.

✓ 13. Reference list (documents listed are cited in text).

Appendixes
__ 14a. Included (they are necessary explanatory information).
__ 14b. None prepared.

Signature of individual completing checklist.

[illegible]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS: NONE
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Georgia Institute of Technology; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: ARMY RESEARCH INSTITUTE OFFICE OF BASIC RESEARCH
6c. ADDRESS (City, State, and ZIP Code): School of Information and Computer Science Atlanta, GA 30332	7b. ADDRESS (City, State, and ZIP Code): 5001 EISENHOWER AVENUE ALEXANDRIA, VA 22333-5600
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: ARMY RESEARCH INSTITUTE; 8b. OFFICE SYMBOL (If applicable): PERI-BR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: MDA 903-86-C-173
8c. ADDRESS (City, State, and ZIP Code): OFFICE OF BASIC RESEARCH 5001 EISENHOWER AVENUE ALEXANDRIA, VA 22333-5600	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 6.11.02.B; PROJECT NO. 2Q1611 02B74F; TASK NO.; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Problem Solving and Learning in a Natural Task Domain

12. PERSONAL AUTHOR(S)
Janet Kolodner/Lawrence Barsalou

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
INTERIM	FROM 9/87 TO 9/88	1988 September	81

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Based on work done in Year 1 of the contract analyzing protocols of students solving diagnostic problems, work in Year 2 of the contract has taken two directions: the creation of AI simulation models to explain several of the learning processes used by students and the creation of an experimental tool and formulation of experiments to find out more about how people learn during problem solving and instruction. This report is divided into two sections. In the first section, the experimental tool and experiments are discussed. In section two, beginning on page 27, some of the AI simulation models are presented. (SR)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: JUDITH ORASANU	22b. TELEPHONE (Include Area Code): (202) 274-8722; 22c. OFFICE SYMBOL: PERI-BR

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Problem Solving and Learning in a Natural Task Domain
Interim Report
September, 1988
MDA-903-86-C-173

Janet Kolodner
Lawrence Barsalou
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Based on work done in Year 1 of the contract analyzing protocols of students solving diagnostic problems, work in Year 2 of the contract has taken two directions: the creation of AI simulation models to explain several of the learning processes used by students and the creation of an experimental tool and formulation of experiments to find out more about how people learn during problem solving and instruction. This report is divided into two sections. In the first section, the experimental tool and experiments are discussed. In section two, beginning on page 27, some of the AI simulation models are presented.

1 Introduction

The goal of this three year project is to explain the details of some of the problem solving and learning processes employed by novice problem solvers as they become more expert. In particular, we are interested in the effects of individual problem solving and learning experiences on later problem solving.

In year 1 of the project, we collected and analyzed protocols of students solving several sequences of diagnostic problems in the domain of car mechanics. A theoretical analysis of these protocols led us to several working hypotheses that we have based our second-year work on.

We found three types of knowledge necessary for diagnosis.

- **Qualitative reasoning rules** provide knowledge about what system behaviors derive from other system behaviors or states.
- **Symptom-fault rules** provide associational knowledge that associates symptoms and other contextual factors with potential faults.
- **Reasoning strategies** provide meta-knowledge about what actions to take in solving a problem.

In learning these three types of knowledge, a student learns two types of descriptive knowledge about the device it is learning about: how the system works and how it malfunctions. This is traditionally called the students *mental model* of the device. The student also learns how to use that knowledge to solve problems or troubleshoot.

Based on these findings from the first year, second year work has gone in several directions. From the psychology side, the year was spent on several tasks:

- building an instruction tool called MECH that can be used to run experiments to find out more detail about what people are learning and what instructional methods work best in teaching those things,
- continuing the theoretical analysis of the first year in an attempt to describe it more rigorously, and in particular, to come up with a set of dimensions useful for assessing observed learning,
- deriving experimental techniques and experiments that will allow us to rigorously describe both the learning students do and the contexts in which they do it.

MECH is an especially important part of the work done in year two of the contract. As an experimental tool built to record the results of experiments in a teaching environment, it has the potential to serve several functions:

- It provides a simulation environment for problem solving, including graphics and help facilities. Thus, with the right knowledge in it, it could be used by students to practice what they have learned without the need for the particular device they have learned about being available.

- It provides an environment for teaching. It has facilities for providing feedback, for providing explanations to students, and for choosing problems to work on. It could therefore be used as a teaching tool.
- It provides an environment for experimentation. It records key strokes and keeps track of latency times. It also allows for different kinds of teaching/learning situations to be set up, thus allowing an experimenter to evaluate the differences between several different teaching strategies.

Work on the AI side of the project has also been in three areas:

- an in-depth investigation of *learning by understanding explanations*, a learning process in which the student integrates what the teacher presents into his/her current mental model,
- investigation of the case-based reasoning processes employed during learning, and
- the creation of memory models that integrate the three different kinds of knowledge problem solvers use and that support both learning processes that we have been investigating.

Work in AI distinguishes itself from other work on machine learning by focussing on the learning that happens in real situations in conjunction with a non-ideal teacher. The description of work done in AI focuses on our investigation of learning by understanding explanations (LBUE), with description of the other two investigations when necessary to explain LBUE processes.

The report has two sections. The first describes work done in psychology. The second, beginning on page 27, describes work done in AI.

2 Learning and Instruction

Barsalou and Hale have focused their efforts on three projects. First, they developed a computer interface, MECH, for studying the learning and troubleshooting of mental models. Second, from reviewing the literature and building MECH, they developed theoretical analyses for viewing the learning and troubleshooting of mental models. Third, they mapped out a space of techniques for studying the learning and troubleshooting mental models. Recently, they have begun using these techniques in an experiment on learning, and they plan to perform additional experiments on troubleshooting in the coming year. We discuss each of these projects next in turn.

2.1 The MECH Computer Interface for Learning and Troubleshooting Mental Models

We spent the past year developing MECH, which is a computer interface for learning and troubleshooting mental models. MECH is currently set up for learning and troubleshooting small gasoline engines. However, the program has been written so it can support any kind of mental model (e.g., electronic circuits, power plants). According to our programmer, Mike Cox of Microspheres, few changes to the program would be necessary to handle another model besides small gasoline engines. Instead MECH is essentially a syntactic engine that can handle materials for any kind of model. To present a different model, all that needs to be done is construct different graphics and text files for MECH to process. This is no small task, given we spent much time this year developing these files. It is well-known in the tutoring business that the most of the effort in developing a tutor goes into coding expert knowledge, and that is what we spent much time doing. However, the MECH program is also quite extensive, given it has taken our programmer a year to construct it.

MECH has an extensive parameter structure, which allows flexibility in configuring it for particular experiments. Parameter setting has been streamlined so that calling a single parameter file suffices for setting all the parameters. MECH also allows editing and saving new parameter files easily and quickly.

MECH consists of six modules that interact to support learning and troubleshooting a mental model. They are:

1. HELP
2. MOVE
3. TUTOR
4. JOBS
5. TESTS
6. REPAIRS

The top-level menu provides access to these modules at any point. Nested menus provide further options within these modules. We discuss these modules in turn.

2.1.1 HELP

The help facility contains instructions on how to use each module. The instructions are sufficiently detailed so that someone having no experience with MECH could completely learn how to use it. (Actually, these instructions don't describe how to configure MECH for particular experiments, since we don't want subjects having access to configuration parameters.) Instructions in the help facility contain numerous examples, often graphically depicted, to insure clarity. Instructions can be easily changed at any time by changing files. Given these files are parameters in MECH parameter files, different experiments with MECH can use different instruction files. As a result, instruction files can be used to implement various learning and troubleshooting manipulations.

2.1.2 MOVE

MECH assumes that the device being modeled can be decomposed hierarchically into subsystems. In small engines, for example, the engine is the largest system, decomposing into subsystems such as the ignition system, fuel system, and drive train. These subsystems in turn often decompose into more specific subsystems. For example, the ignition system decomposes into the magneto and breaker points. Eventually all subsystems decompose into terminal components. For example, the breaker points decompose into terminal components such as the stationary point and moving point.

MECH contains a graphical depiction of every subsystem–it does not contain graphs for terminal components. MECH currently contains graphs for the engine as a whole and for subsystems such as the ignition system, magneto, breaker points, and so forth. These graphs are organized hierarchically, such that each subsystem is nested in the larger subsystem to which it belongs. Note that strict hierarchical structure is occasionally violated. Sometimes a component bridges two subsystems such that it can be conceived of as belonging to both. In small engines, for example, the spark plug belongs to both the ignition system and the cylinder assembly. In these cases, we show the component in the graphs for both subsystems. For small engines, MECH currently contains 19 hierarchically nested graphs. However, MECH is designed to handle any number of graphs in any hierarchical configuration.

Every graph contains three kinds of information:

1. COMPONENT INFORMATION. A component of a graph can be either a terminal component or a more specific subsystem. At high levels in the hierarchies, most components may be subsystems, having graphs themselves.

For example, the graph for the fuel system contains components for the fuel pipe and carburetor, both of which are subsystems and have their own graphs. The lowest-level graphs contain only terminal components. Components are always associated with verbal labels that identify them.

2. INTERNAL INPUT-OUTPUT RELATIONS. These relations represent the inputs and outputs between components in the subsystem. In the graph for the fuel system, an internal relation shows that fuel flows from the fuel intake pipe to the carburetor. Internal relations are always associated with verbal labels that identify them.

3. EXTERNAL INPUT-OUTPUT RELATIONS. These relations represent inputs entering the subsystem and outputs leaving the subsystem. In all cases, these relations connect to a component in the subsystem and then point in or out of the subsytem. For example, an external input to the cylinder assembly is air-fuel mixture from the fuel system; an external output from the cylinder assembly is displacement from the connecting rod to the drive drain. External relations are always associated with verbal labels that identify them.

At higher levels in the hierarchy, spatial relations between components are sometimes violated in the graphs, given this information is hard to convey graphically and clearly (e.g., for the ignition system). At lower levels, spatial relations are often preserved (e.g., for the spark plug). In other words, at lower levels the graphs often look a lot like their physical counterparts.

The MOVE module allows subjects to move through the space of graphs. Except in a few situations, a graph is alw ys present on the screen. To move to a new graph, subjects call MOVE and implement one of its functions. These functions include moving to the root graph for the engine, moving up one level in the hierarchy from the current graph, or moving down to a lower subsystem. Subjects can quickly and clearly view decomposition of the engine using these three "vertical" functions. Two further functions allow traversing external input and output relations. By tracing back through the inputs to a subsystem, subjects can discover how various forces and substances converge on that subsystem. Following the outputs out of a subsystem allows subjects to see how the current subsystem affects other subsystems. Subjects can quickly and clearly view the functionality of the engine using these two "horizontal" functions.

Using MOVE is essential to the TUTOR, TEST, and REPAIR modules. Tutoring, testing, and repairing at any point during program use are always limited to the current graph. For example, if a subject is currently viewing the graph for the breaker points, he or she can only learn about, test, and repair the breaker points.

To learn about, test, and repair some other part of the engine, the subject must move there first.

An advantage of this design is that it allows us to track what parts of the engine are currently relevant to the subject. Because MECH records all keystrokes, we can follow a subject's path through the model and see when they needed tutoring, when they ran tests, and when they attempted repairs. Because we also store the latency for each keystroke, we can see how much time a subject spent at each point in the model performing an operation.

MECH is also designed such that the keystrokes provided by one user can be used to control MECH for other users. For example, an experimenter could move through MECH performing breadth-first search for a fault. A subject could later interact with MECH while it performs the exact search performed by the experimenter. Similarly, an experimenter could present information from the tutor only about system level information as opposed to component level information. In general, we use this keystroke control facility to implement various instructional manipulations.

2.1.3 TUTOR

The graphs provide some tutoring, given they represent the components and relations for each subsystem. However, the tutor, through text, describes these components and relations in much greater detail. Analogous to the graphs, the text units are organized hierarchically. The root contains the text describing the engine as a whole at a very general level. Texts associated with subordinate nodes of the engine topology describing increasingly specific subsystems that constitute decomposition of the engine.

For a given subsystem, subjects can access one SYSTEM DESCRIPTION of the subsystem as a whole and one COMPONENT DESCRIPTION for every component in the subsystem. The system description provides a general description of how the subsystem functions as a whole and how it interacts with external subsystems. A component description describes the structure and local function of a specific component in the current graph. As discussed in the previous section, we can implement keystroke control during tutoring such that we control which descriptions the subject receives in what order. Alternatively, we can allow subjects free reign in accessing these descriptions.

2.1.4 JOBS

Once subjects have learned how to use MECH from HELP and know something about small engines from MOVE and TUTOR, they can begin practice at troubleshooting. To make their first repair, they access a jobs menu, which presents in-depth information about the first/next job. This job description may include the owner of the engine, the type and model of the engine, the age of the engine, the engine's maintenance history, the engine's repair history, and comments from the owner about observed symptoms and possible faults. There is much flexibility in exactly what kinds of information a subject receives.

Across problems, we can correlate various features of the job description with faults to look at how subjects form generalizations about troubleshooting (e.g., whenever the engine is a Tecumseh more than 3 years old, the problem is likely to be in the electrical system; whenever there is smoke coming out of the exhaust, it is likely that air intake into the carburetor is clogged). We can also look at remindings by seeing whether the features of the current job cause subjects to look for a fault that occurred in a previous repair having the same features.

JOBS also allows subjects to review previous problems they solved. One menu option allows reviewing the jobs in order, listed by name. Another option allows viewing specific pieces of information from a given job. Because subjects have to ask specifically for each piece of information about a previous job, we are able to keep track of exactly what they're interested in (through their keystroke file). This will be useful assessing how subjects are generalizing, given they will probably want to look up previous information in order to see if a generalization is warranted. For example, a subject might want to look back over previous problems for which the engine was only three months old to see if all engines were of the same type and exhibited the same fault.

2.1.5 TESTS

Once subjects have read the description for the first/next job, they are free to perform various tests on the engine's subsystems and components. A subject's goal is to find and repair all faults in the current engine being repaired, where a fault can be a malfunction in any subsystem or component (faults are preset by a parameter file read in for the problem). Only one fault may exist, or there may be several. When there is more than one fault, they may be related or unrelated, in the same subsystem or in different subsystems. At any given time, the engine may be in either an on or off state, since some tests require that the engine be on. If the current faults preclude the engine being on, the subject will not be able to turn it on to run these tests.

Whenever a subject performs a test, he or she receives feedback about the results. The test may have been unnecessary, or it may have any number of other possible outcomes. The outcome of a test depends on PRODUCTIONS associated with it in the current parameter file for the problem. Each production has a set of BROKEN CONDITIONS and a set of WORKING CONDITIONS that trigger it. Broken conditions are engine components that must be broken for the production to fire (i.e., faults that haven't been repaired yet). Working conditions are engine components that must be working for the production to fire (i.e., faults that have been repaired). If both sets of conditions are completely met, the message associated with the production is presented to the subject as the outcome of the test. One advantage of this approach is that the outcome of a given test can vary widely, depending on the current state of other subsystems and components in the engine. For example, imagine that there are obstructions in the carburetor, both for air and fuel intake. Further imagine that a subject tests the cylinder to see whether it is admitting air-fuel mixture. If neither the air or fuel intake has been fixed, subjects receive a message saying that nothing is entering the cylinder; if the air problem has been repaired, subjects receive a message saying that only air is entering; if the fuel problem has been repaired, subjects receive a message saying that only fuel is entering; if both have been repaired, subjects receive a message saying that the test revealed no fault.

As can be seen from this example, a test can report that a component is not working properly even though it is not a fault (i.e., the intake value of the cylinder may report not receiving air-fuel mixture, not because the valve is broken, but because other components connected by external relations are broken). By taking advange of such relations, we can orient subjects toward causal reasoning about symptoms and faults. Moreover, because a given problem takes both broken and working conditions into account, it provides a dynamic testing environment in which the outcomes of tests vary as a function of the repair process.

2.1.6 REPAIRS

A subject is free to make any repair at any time, regardless of whether it needs to be done. If the repair was necessary, subjects receive a message saying that the repair was needed. If the repair was unnecessary, subjects receive feedback saying the repair was unnecessary. When a repair is made, the fault disappears, thereby changing which test productions will fire. To see if the current engine has been completely repaired, the subject can try to start the engine at any time. If no faults remain, then the engine has been repaired, and the subject is asked to go on to the next problem.

2.1.7 PAYOFFS

A cost is associated with every test and repair. Subjects receive constant information about how much tests and repairs cost and how much cost they have incurred cumulatively for the current job. Once a job is finished, subjects receive information about the the optimal cost and the actual cost incurred. The payoff structure is set up such that the closer subjects get to the optimal costs, the more money they can make by being in the experiment. This will encourage subjects to learn two kinds of information.

1. To the extent that subjects learn the structure and function of the engine, they will be able to identify faults optimally. Because symptoms in the job description may be causally related to the faults, and because multiple faults may be causally related, subjects' understanding of engine subsystems and operation will help them reason through problems as quickly as possible.
2. To the extent that subjects learn arbitrary correlations between irrelevant (non-causal) features in the job description and faults, they will be able to predict faults quickly upon initially reading about the job.

Consequently, MECH strongly encourages subjects to learn and discourages subjects from using strategies such as breadth-first and depth-first search, which typically are quite costly. Of interest will be the ways in which subjects utilize remindings and generalizations to support optimal fault prediction. We can also use payoff information to measure troubleshooting skill. To the extent subjects are good at troubleshooting, their actual costs should approximate the ideal costs. As discussed later, we plan to use this measure to see what kinds of learning conditions produce the best skill at troubleshooting.

2.1.8 CURRENT STATUS

The version of MECH described here is nearly finished. We have a working version up and running. However, there are several bugs that need to be handled and a few minor features that need to be added. We anticipate having MECH fully completed in about two weeks. Once it is, we will spend the next month testing it and streamlining its user interfaces. We will also be developing problem sets that explore various issues in causal reasoning, generalization, and reminding. All of this preparation should be done by January 1, 1989, at which point we will begin running pilot experiments on troubleshooting.

2.2 Theoretical Analysis

In this section we present theoretical analyses that we have performed in the process of developing MECH and reading the literature. The principles in this section structure the design of MECH and our experimental work. For these reasons, we present our analyses in some detail.

2.2.1 Analogical Models

We have found that the following representational assumptions and entites go a long way in accounting for the structural aspects of mental models. First, any mental model is based on the following three representational assumptions:

1. ANALOGICAL MAPPING: There is generally a 1:1 mapping of components and relations in a physical device to representations of components and relations in a mental model (Johnson-Laird, 1983).
   - CAVEAT 1A: Not all components and relations in a physical device may be represented in a mental model.
   - CAVEAT 1B: Components and relations may not be correctly represented.
2. HIERARCHICAL ORGANIZATION: To the extent the organization of a physical device is hierarchical, the organization of a mental model may be hierarchical. More specifically, the device and the mental model both decompose to subsystems, which in turn decompose to more specific subsystems, etc., before decomposing to terminal components. As discussed earlier, components belonging to two subsystems may occasionaly violate the strict hierarhical organization of components. Even with these violations, it is still possible to decompose a device in a quasi-hierarchical manner that serves as a useful organization of components. In a small engine, decomposition may procede from the engine to the ignition and fuel systems, from the ignition system to the magneto and spark plug, from the magneto to the breaker points and coil, from the breaker points to terminal components such as the stationary point and moving point.
3. MULTIPLE MODELS: A given mental model may be one of many possible for the same physical device. There are many ways models can differ. There can be be simple ways in which different people represent the same device in slightly different ways, with there being more similarity than dissimilarity between models. On the other, there may be multiple models for the same

device that capture fundamentally different kinds of I/O relations and serve fundamentally different kinds of reasoning goals (e.g., White and Fredericksen, 1986).

Any hierarchically-organized mental model contains the following representational entities:

- COMPONENTS: These include representations of the specific subsystems that compose a more general subsystem, as well as the terminal components of the most specific subsystems. In a small engine, components of the engine include the fuel system and ignition system (decomposition of a general subsystem to more specific subsystems); components of the coil include the primary wire and secondary wire (decomposition of a subsystem to terminal components).

- INTERNAL RELATIONS: These are representations of the input/output relations between components of a subsystem. In the ignition system, these include passage of current from the magneto to the breaker points to the spark plug; in the fuel system, these include passage of fuel from the tank to the fuel pickup pipe to the carburetor.

- EXTERNAL RELATIONS: These are representations of input/output relations from one subsystem to another. External relations are only possible when a hierarchical decomposition of a mental model exists. Note that external relations in some sense provide violations of hierarchical structure, since they criss-cross the decomposition hierarchy in ways that violate class inclusion. In a small engine, these include passage of current from the ignition system to the cylinder assembly and displacement of the drive train by the cylinder assembly.

Another kind of entity related to mental models seems quite interesting and important. It is similar to what Murphy and Medin (1985) and Schank, Collins, and Hunter (1986) have argued structures categories:

- GENERAL PHYSICAL MECHANISMS: These are representations of general physical mechanisms that in some way help integrate (at least initially) the components and relations of mental models. Examples of such mechanisms include the generation and amplification of electrical energy, the combustion of air-fuel mixture, the build-up and dissipation of heat, lubrication, and so forth. One way to think about these mechanisms is as very abstract input/output relations between very abstract components. Once these mechanisms are understood, they become mapped into more specific applications to help produce mental models. An interesting question is whether acquiring them can should precede acquiring a mental model for a physical system or whether they are more easily acquired after having gained knowledge of how at least one physical system works. It is possible that the acquisition of a new mechanism may cause a person to fundamentally restructure a mental model (Collins, Salter, & Tenney, 198?).

2.2.2 Processing Mental Models

We have found that three kinds of rules go a long way in accounting for the processing of mental models. They are:

1. QUALITATIVE REASONING RULES: These rules specify how the outputs from one component determine the outputs of another. People use these rules to simulate performance of the device. Series of these rules may be applied to see how an input to one component produces effects over paths of relations that emanate from the component. Following Hegerty et al. (1988), the operation of these rules depends on the properties of the the respective components, as well as on the inputs they receive (e.g., the rate of fuel flow through a tube depends on its diameter, as well as on the amount of fuel it receives as input). These properties and inputs provide constraints on the behavior of components. Qualitative reasoning rules capture these constraints and allow qualitative prediction about performance. More specific forms of these rules may allow quantitative prediction.

2. SYMPTOM-FAULT RULES: These rules start with observed problematic symptoms and provide hypotheses about what components might be at fault. For a small engine, a symptom-rule might state that whenever there's a strong gas smell during engine operation, check to see if the choke is broken. Another rule might state that whenever the engine type is Briggs and Stratton, check to see if the condenser is broken.

3. META-RULES: These include rules about how the topology of a mental model should be searched to find a fault (e.g., breadth-first versus depth-first); rules about the transitivity of qualitative reasoning rules (e.g., if component X produces an input to component Y, and if component Y produces an input to component Z, then X produces the input to Z); rules about how to handle remindings; etc. In contrast to qualitative reasoning and symptom-fault rules, meta-rules may be fairly domain independent. In general, meta-rules guide the

executive control of troubleshooting by setting goals, deciding how to handle errors, handling interuptions and unexpected results, etc. (cf. Norman & Shallice, 1986).

An organizational principle also seems important to processing:

- COMPILATION: With practice at using any sequence of rules repeatedly, the sequence may become compiled into a procedure that produces more efficient processing in the future. Sequences of qualitative reasoning rules may become automated for frequent kinds of qualitative reasoning. Sequences (or simultaneous sets) of symptom- fault rules may become automated to zero in quickly on suspected faults. Sequences of meta-rules may become automated to minimize wasted resources and non-optimal behavior. Moreover, combinations of different types of rules may become automated to the extent they frequently occur in a systematic pattern.

2.2.3 Learning Mental Models

There appear to be two important kinds of learning that can occur for mental models: (1) learning a mental model; (2) learning to use a mental model for troubleshooting a physical device. Learning mental models is addressed in this section; learning how to troubleshoot is addressed in the next.

Learning a mental model can be assessed on the following six dimensions:

1. ACQUISTION OF COMPONENTS: To what extent are the components of the physical device represented in the mental model?
2. ACQUISITION OF HIERARCHICAL STRUCTURE: To what extent is the hierarchical organization of components in a physical device represented in the hierarchical organization of components in the mental model?
3. ACQUISITION OF INTERNAL RELATIONS: Assuming components are hierarchically organized in a mental model, to what extent are the components within a particular subsystem integrated by the appropriate input/output relations?
4. ACQUISITION OF EXTERNAL RELATIONS: Assuming subsystems are hierarchically organized in a mental model, to what extent are they integrated by the appropriate input/output relations?

5. ACQUISITION OF GENERAL PHYSICAL MECHANISMS: To the extent that general physical mechanisms are important to properly integrating the components of a mental model, to what extent are these mechanisms represented and integrated with the model?

6. ACQUISITION OF MULTIPLE MENTAL MODELS: To the extent that fundamentally different models can usefully represent the same device, to what extent have they been represented and integrated?

2.2.4 Learning to Troubleshoot

Learning how to use a mental model during troubleshooting can be assessed on the following five dimensions:

1. USE OF REMINDINGS: To what extent do people use previous problem solving episodes to solve new problems (Ross, 1984)? The natural contrast is: To what extent do people use meta-rules such as breadth- first search to solve problems? Related issues include: What characteristics of the current problem trigger a reminded episode? What information about a previous problem gets stored in an episode? What information is utilized from an episode? How is this information used in the current problem? What kind of generalization takes place as a result?

2. ACQUISITION OF SYMPTOM-FAULT RULES: To what extent do people develop symptom-fault rules? Are they generalized from a single episode or from several? If several, how many? What information do they keep? What information do they throw away? Note that other information besides symptoms may constitute the triggering conditions of these rules (e.g., kind of engine, maintenance history, etc.) How are symptom-fault rules related in memory to the episodes that produced them?

3. ACQUISITION OF QUALITATIVE REASONING RULES: To what extent have rules been acquired that allow accurate simulation with the model? To the extent these rules are present, accurate prediction of how the system will behave should be possible given an input. Of interest is whether predictions about normal functioning are more accurate than predictions about how the engine functions when broken.

4. ACQUISITION OF META-RULES: To what extent do people develop various high-level rules and strategies such as breadth-first search to support troubleshooting? To what extent are meta-rules created for troubleshooting

the current device, versus being adapted from some other domain? What aspects of the current troubleshooting create or modify these rules?

5. FURTHER ACQUISITION OF THE MENTAL MODEL: To what extent do people continue developing knowledge of the mental model during troubleshooting? All six kinds of learning described in the previous section could occur. To the extent such learning takes place, what conditions promote it?

2.3 Experimental Techniques for Studying the Learning and Troubleshooting of Mental Models

2.3.1 Measuring the Acquisition of Mental Models

This section proposes specific ways to study the acquisition of mental models. The next section proposes ways to study the acquisition of troubleshooting.

Earlier we presented six kinds of learning that could occur for mental models: acquisition of components, hierarchical structure, internal relations, external relations, general physical mechanisms, and multiple mental models. In the context of using MECH, we are not currently able to study the acquisition of general physical mechanisms or multiple mental models (although MECH could be made to handle these with a moderate amount of effort). Consequently, we are only in a position to address the acquisition of components, hierarchical organization, internal relations, and external relations. Arguably, however, these are the most basic aspects of mental models to study.

1. TIMECOURSE ISSUES. There are four timecourse issues that must be considered in assessing acquisition. They are:

   (a) RELATIVE ACCRUAL OF COMPONENTS AND RELATIONS: What kinds of information accrue early in the acquisition of a mental model? What kinds of information come in late? Do components generally precede internal relations? Do external relations generally precede internal relations? Some theories make these predictions. On the other hand, these various types of information may accrue at relatively equal rates. Another issue concerns the heigth of information in the topology of the engine. Are components and relations from higher-level subsystems acquired faster than components and relations from lower-level subsystems?

   (b) RELATIVE ACCRUAL OF ORGANIZATION: How does the organization of a mental model change over time? Are early models relatively unhierarchical, with later models becoming increasingly differentiated according to subsystems? Or are early models primarily organized hierarchically in terms of components, with later models become organized more functionally by internal and external relations?

(c) RELATIVE LOSS OF COMPONENTS AND RELATIONS: After a model has been acquired, which kinds of information are retained the longest and which are forgotten most rapidly? For example, are components better remembered than relations? Are external relations remembered better than internal relations? Are components that are involved in more relations better remembered than components involved in fewer relations? Is information higher in the hierarchy better remembered than information lower in the hierarchy?

(d) RELATIVE LOSS OF ORGANIZATION: To what extent is organizational information lost over time? What kinds of organizational information are best remembered? Is loss of organizational information dependent on hierarchical height? Is organizational information lost more rapidly than component information?

2. EXPERIMENTAL TECHNIQUES. The following experimental techniques can be used to address these questions:

   (a) MEASURING MENTAL MODELS AT VARIOUS POINTS IN LEARNING. Subjects could receive multiple tutoring sessions on a physical device, with the same material being presented in each session (i.e., utilizing MECH under experimenter control). At the end of each session, we could assess a subject's current model with various recall and recognition measures. By coding responses with respect to components, internal relations, external relations, organization, and hierarchical height, we can assess how much of each kind of learning occurred. By comparing these measures across sessions, we can see what kinds of learning occur early versus late during acquistion of the model. In other words, we can track the timecourse of learning for these various aspects of the mental model.

   Another way to do this study would be to allow subjects to use MECH however they wish to tutor themselves (i.e., utilizing MECH under subject control). We could stop subjects at various points and assess their memory for the different kinds of information. Subjects could return for subsequent sessions of a similar type. By also seeing how subjects search through the tutor to learn, we can also get a sense of the kinds of information they want to see early in learning versus the kind of information they want to see late.

(b) MEASURING MENTAL MODELS AFTER VARYING DELAYS. After subjects have achieved some criterial level of learning, we can test them after varying delays (e.g., immediately, 1 day, 1 week, 2 weeks, 4 weeks, 3 months, 6 months, 1 year). This could be done both between and within subjects. Between-subjects testing would allow the best assessment of what kinds of information are lost at what rate over time. Within-subject testing, in conjunction with between-subject testing, would allow assessing the extent to which testing maintains the mental model in memory. More specifically, subjects could be tested after 1 week, again after 1 month, again after 6 months, and again after 1 year. Of interest would be seeing how their forgetting at each point in time compared to the forgetting of the comparable between-subjects group. Another useful manipulation would be to run a third condition in which subjects received tutoring instead of testing. More specifically, subjects could be tutored again after 1 week, again after 1 month, again after 6 months, and again after 1 year. Of interest would be seeing how their forgetting at each point in time compared to the forgetting of the comparable within-subjects, testing group. Does subsequent tutoring or testing best maintain a mental model in memory?

(c) MODERATION BY LEARNING CONDITIONS. Actually, assessing the accrual and loss of information in mental models may be moderated by learning conditions. For example, some learning conditions may optimize the learning of components, others may optimize the learning of relations, etc. Several examples of learning conditions are:

- LEARNING COMPONENTS FIRST WITHOUT LEARNING RELATIONS OR ORGANIZATION
- LEARNING INTERNAL RELATIONS WHILE LEARNING COMPONENTS
- LEARNING THE HIERARCHICAL ORGANIZATION OF COMPONENTS WHILE LEARNING COMPONENTS
- LEARNING HIERARCHICAL ORGANIZATION AND EXTERNAL RELATIONS WHILE LEARNING COMPONENTS

The central issue in the above learning methods concerns the proper mix of information to give subjects at various points in learning. Does compartmentalizing information lead to faster or slower learning, and does it lead to better or poorer memory? Or does mixing various types of information optimize learning and memory? If mixing is better, then what are the optimal mixes? Over what timecourse? The basic way to answer these questions is simply to manipulate learning conditions and observe the effects on learning and retention. Another interesting learning mode is:

- LEARNING ABOUT COMPONENTS, RELATIONS, AND ORGANIZATION IN THE CONTEXT OF TROUBLESHOOTING

Subjects may acquire information about a mental model faster if they are trying to troubleshoot it then if their task is simply to memorize the material. This kind of learning may also produce the best retention of mental models. One problem is that the haphazardness of problem solving may make it difficult for subjects to receive systematic and exhaustive coverage of the engine's structure and function. Consequently, a mix of troubleshooting and tutoring may be optimal.

2.3.2 Measuring the Acquisition of Troubleshooting

Earlier we presented four kinds of learning that could occur during troubleshooting: remindings, symptom-fault rules, qualitative reasoning rules, and meta-rules. In the context of using MECH, we are currently able to study all four.

The following nine paradigms may provide various insights into the role that remindings, symptom-fault rules, qualitative reasoning rules, and meta-rules play in the development of troubleshooting expertise:

1. THE DISTRIBUTION OF STRATEGY TYPES OVER THE DEVELOPMENT OF EXPERTISE
2. FACTORS THAT DETERMINE REMINDINGS
3. FACTORS THAT DETERMINE THE USE OF SYSTEM-FAULT RULES
4. FACTORS THAT DETERMINE THE USE OF QUALITATIVE REASONING RULES
5. FACTORS THAT DETERMINE THE USE OF META-RULES
6. THE EFFECT OF MENTAL MODELS ON TROUBLESHOOTING
7. THE EFFECT OF TROUBLESHOOTING ON MENTAL MODELS
8. THE ORGANIZATION OF TROUBLESHOOTING EPISODES IN MEMORY
9. THE CONTENT OF EPISODES AND SYMPTOM-FAULT RULES.

Each of these nine paradigms is discussed next in turn.

1. THE DISTRIBUTION OF STRATEGY TYPES OVER THE DEVELOPMENT OF EXPERTISE. The basic question is: To what relative extents do subjects use remindings, symptom-fault rules, qualitative reasoning rules, and meta-rules over the development of expertise. Do novices initially use meta-rules such as breadth-first and depth-first search to find faults? Do they sometimes use qualitative reasoning to map symptoms onto possible faults? After subjects have performed a number of troubleshooting episodes, do they start using remindings to guide search? Once they have been reminded a few times, do they start using symptom-fault rules that are generalizations of remindings? After subjects have acquired symptom-fault rules, what do they do upon receiving a problem that bears no resemblance to a previous problem? Are they more likely now to use qualitative reasoning, or do they fall back on meta-rules? Does the likelihood of qualitative reasoning increase or decrease with experience?

   We can study these questions by giving subjects problem sets of 100 problems over several hours, for example, where each problem presents a broken engine with one or more faults. By manipulating the similarity of job characteristics across problems (i.e., customer, engine type and model, symptoms, maintenance history, previous repairs, customer observations, and faults), we can create conditions that will produce remindings and generalizations. We can identify the use of various strategies using the following techniques:

   (a) META-RULES. If subjects are using meta-rules such as breadth-first and depth-first search, then we will be able to determine this by the pattern of their keystrokes (which are stored completely during troubleshooting). Subjects using breadth-first search, for example, should test all the highest-level systems first before proceeding to more specific subsystems and terminal components.

   (b) QUALITATIVE REASONING RULES. If subjects are using qualitative reasoning rules, then we should see two sorts of patterns in their keystrokes. First, if subjects draw qualitative inferences from symptoms to faults, we should see them go directly to the correct fault without going through something like breadth-first or depth-first search. For example, the symptom "strong smell of gas while engine is running" is connected by various qualitative rules to the choke, throttle, and air intake. If the subject immediately tests these components, we can assume they are pursuing qualitative reasoning of a sort. Second, subjects may use the outcome of a test to direct search, rather than continuing with a meta-rule. For example, if a subject discovers that no fuel is reaching the cylinder and that the intake valve is not broken, then the subject may reason that something in the carburetor must be clogged or broken.

(c) REMINDINGS. If subjects are using remindings, then upon receiving a job that shares characteristics with one previous job, they should immediately test the component(s) at fault in the previous job. They should not use a meta-rule or qualitative reasoning to determine search.

(d) SYMPTOM-FAULT RULES. If subjects are using symptom-fault rules, then upon receiving a job that shares characteristics with two or more previous jobs, they should immediately test the component(s) at fault in the previous job. It is essential to note that subjects could be using remindings at this point. In general, it is difficult if not impossible to discriminate exemplar from generalization models, empirically speaking. There may be a fundamental indeterminacy problem here that can not be resolved, much like the indeterminacy problems for serial versus parallel processing and imaginal representations versus propositional representations. Nevertheless, we may discover that there are ways to differentiate these accounts, and we will attempt to do so. Otherwise, we will probably make a theoretical assumption that subjects who show learning after receiving two or more problems of a particular type have extracted a symptom-fault rule. Issues concerning the induction of symptom-fault rules are addressed in a later section.

In summary, we will use patterns in subjects' keystroke files, in conjunction with our knowledge of the problems they have received, to assess their strategies in locating faults. Again, our primary interest will be to see the relative extent to which these strategies are used over the development of expertise.

2. FACTORS THAT DETERMINE REMINDINGS. Basic questions include: To what extent must the current job overlap in features with a previous job for the previous job to be reminded? What kinds of features produce the most frequent remindings, holding amount of overlap constant? Are remindings more likely to occur early in learning rather than late? To what extent must a previous job have occurred recently for it to be reminded? Finally, imagine that a previous job (the "target job') is similar to the current job on several features. Further imagine that we vary the extent to which other previous jobs are similar to target job on features different from those shared by the target and current jobs. Does this decrease or increase the probability of the current job reminding the target job? In other words, if the target job is part of a cluster that may have been generalized, how does this affect accessibility of the target?

   We can study all of these questions in MECH by constructing pairs of problems that overlap on certain features. We can manipulate the number of shared features, the type of shared features, the number of intervening problems, and whether the pair occurs early or late in learning (holding the number of intervening problems constant). We can also manipulate the similarity of previous problems to the target. In all cases, we can assume reminding has occurred if a subject first tests the same component that was at fault in the target problem.

3. FACTORS THAT DETERMINE THE USE OF SYSTEM-FAULT RULES. Basic questions include: To what extent must two or more jobs overlap in features for a symptom-fault rule to be constructed? What kinds of features are most likely to produce a symptom-fault rule, holding amount of overlap constant? How does the probability of forming a symptom-fault rule increase with the number of similar jobs? Are symptom-fault rules more likely to develop late in learning rather than early in learning? Are massed or distributed episodes more likely to produce symptom-fault rules? When a symtpom-fault rule is formed, what information is extracted? Only information that can be related to the fault by qualitative reasoning? Or irrelevant information as well? If irrelevant information is included, is it just as likely to trigger the symptom-fault rule as is relevant information? Finally, how are symptom-fault rules related in memory to the episodes that produced them? Are they clustered together such that activating the symptom-fault rules also activates the episodes? Or are they stored separately?

   We can study all of these questions in MECH by constructing sets of problems that overlap on certain features. We can manipulate the number of shared features, the type of shared features, and the number of problems sharing features. We can manipulate whether similar jobs are massed or distributed and whether they occur early or late in learning. We can see whether the conditions for a symptom-fault rule contain irrelevant as well as relevant information by seeing if later problems that only contain the irrelevant information fire the rule. In all cases, we can assume that a symptom-fault rule has been formed if a subject first tests the same component that was at fault in the previous problems that produced the rule. Again it is important to note the difficulty of disciminating pure exemplar accounts from generalization accounts. Whether a symptom-fault rule is stored with its episodes is addressed in the paradigm that addresses the organization of episodes, discussed below.

4. FACTORS THAT DETERMINE THE USE OF QUALITATIVE REASONING RULES. Basic questions include: Does prior training on the structure and function of a physical device transfer to qualitative reasoning during troubleshooting? If a subject first learns internal and external relations from a tutor, does this later facilitate reasoning about how a symptom might be produced by a faulty component? Or about how a faulty component might affect

another component? Perhaps subjects really only learn to reason qualitatively in the process of troubleshooting. If so, then what types of troubleshooting experience best promote the acquistion of qualitative reasoning rules? To the extent problems form predictable clusters and produce predictive symptom-fault rules, do subjects not learn to reason qualitatively? To the extent problems don't have much predictive structure at all, do subjects primarily use meta-rules and forego qualitative reasoning? Does qualitative reasoning primarily develop when subjects are faced with relatively novel problems, where knowing qualitative relations between components can facilitate search? If so, do these problems promote better qualitative reasoning skills than learning about qualitative relations from a tutor?

We can study these problems in MECH in a couple of ways. First, we can vary the kind of tutoring a subject receives before troubleshooting to see if tutoring affects the ability to reason qualitatively. If tutoring can promote this skill, then we should see benefits from some types of tutoring but not others. Second, we can study qualitative reasoning by controlling the composition of the problem set. We can manipulate the amount and type of predictive structure in the problem set to see if these factors determine how well subjects reason qualitatively.

Measuring the ability to reason qualitatively can be done in two ways. First, we can ask subjects specific questions about the relations between two components. For example, "If the magneto is broken, what other components might not function properly." Or, "If the magneto is not working properly but is not broken, what other components might be broken, thereby causing the magnetoto malfunction." Second, we can observe qualitative reasoning indirectly by looking at how subjects search for faults. On some problems, symptoms may be qualitatively related to the faults. If subjects are good at qualitative reasoning, they should make the connection and find the fault quickly. On other problems, we can direct subjects to a component that is not broken but that is not working properly. If subjects are good at qualitative reasoning, they should converge quickly on the faulty component that is making the unbroken component perform improperly.

5. FACTORS THAT DETERMINE THE USE OF META-RULES. Basic questions include: Do subjects generally prefer breadth-first or depth-first search? What conditions promote these preferences? What other general rules do subjects develop to direct problem solving?

   To see whether subjects prefer breadth-first to depth-first search, we will in some cases present them with problem sets that have no predictive structure and see what they do. To see whether different conditions promote these two types of search, we will simply note situations where a particular type of search is preferred. At this point, it is not clear what these situations might be. As far as other general rules, we again don't have any specific hypotheses and will simply be on the look out for systematic patterns that suggest the use of meta-rules.

6. THE EFFECT OF MENTAL MODELS ON TROUBLESHOOTING. The basic question here is: What effect does prior tutoring on a mental model have on troubleshooting? Are subjects any better at troubleshooting after having learned the structure and function of the device to be repaired? If so, then are particular types of tutoring better than others at promoting troubleshooting skill? Does focusing on hierarchical structure during tutoring encourage breadth-first and depth-first search during troubleshooting? Does exposing subjects to all the possible tests and repairs during tutoring lead to better troubleshooting later? If so, does presenting tests and repairs work best when they are presented breadth- first, depth-first, or following paths of qualitative reasoning?

   To assess this question, we will tutor subjects in various ways described in Section III.A and see what kinds of tutoring produce the best troubleshooting. We will also include a condition with no troubleshooting to see if these subjects do as well as tutored subjects. The measure of troubleshooting ability will simply be how quickly subjects find faults (i.e., how closely actual costs approximate ideal costs in the payoff structure). We wıil also look more specifically at the abilities to reason qualitatively, to use meta-rules, and to construct symptom-fault rules, all of which may be affected by various kinds of tutoring.

7. THE EFFECT OF TROUBLESHOOTING ON MENTAL MODELS. The basic question is: What effect does troubleshooting have on learning mental models? Do people learn mental models better in the context of troubleshooting than in a tutoring context? If troubleshooting primarily serves to increase the quality of a mental model, what kinds of changes does it produce?

   We can explore these questions by assessing people's mental models after troubleshooting, using all the same measures described earlier in Section III.A (e.g., knowledge of components, internal relations, external relations, hierarchical organization). If troubleshooting alone produces better learning than tutoring alone (given a constant amount of time spent), then troubleshooting subjects should score higher on these measures than tutoring subjects. To see how troubleshooting changes already established models, we can first tutor subjects on a mental model, then have them perform troubleshooting, and then assess their mental model. We can compare these subjects to others who were tutored but who did not perform troubleshooting. We can then see what kinds of differences exist between the mental models of these two groups on our variety of memory measures.

8. THE ORGANIZATION OF TROUBLESHOOTING EPISODES IN MEMORY. The basic question is: How are episodes integrated with symptom-fault rules and qualitative reasoning rules? If subjects store episodes with the rules that identified their faults, then subjects should later cluster episodes that share a common rule. For example, if several episodes involved the same qualitative reasoning rule, then these episodes should be recalled together. If several episodes involved the same symptom-fault rule, then these episodes should be recalled together. Another possibility is that subjects store episodes according to the topology of the model, with each episode being associated with its fault(s) or with every component that was tested.

   We can assess this issue by asking subjects to recall all previous jobs at the end of troubleshooting. To the extent that subjects have organized exemplars according to particular organizational principles, we should see jobs clustered in those ways.

9. THE CONTENT OF EPISODES AND SYMPTOM-FAULT RULES. The basic question is: What information is recorded in memory for episodes and symptom fault rules. Are memories of episodes biased toward information relevant to finding the fault? Or is irrelevant information remembered well, too? Do symptom-fault rules contain only predictive information? Or do subjects also generalize over irrelevant information that is not predictive?

   To assess these questions, we can ask subjects to recall information about previous jobs and about the conditions of symptom-fault rules. For episodes, we can give subjects enough information to identify a job and then ask them to recall the remaining details. We can further probe their memories by specifically asking them to recall the type of engine, the maintenence history, etc. For symptom-fault rules, we can give subjects a fault that occurred in more than one job and ask them to provide characteristics shared by jobs having that fault. Again we can probe subjects by specifically asking them to recall the type of engine, the maintenence history, etc. Using these recall techniques, we can assess the information stored with episodes and symptom-fault rules.

2.4 Current and Future Experiments

2.4.1 Current Experiment

We are currently in the process of performing our first learning experiment, which had the following three purposes. First, we wanted to compare several modes of learning to see which is optimial. Actually we expected to find that different modes have different strengths and weaknesses, rather than there being a simple rank ordering of modes from best to worst. Second, we wanted to identify the fastest but most effective way to teach subjects about a small engine so that they can start learning about troubleshooting as quickly as possible in later experiments. Third, we wanted to get a general feel for how subjects acquire knowledge about the structure and function of small engines. We thought we might observe some interesting phenomena that would stimulate further hypotheses about how people acquire mental models for physical devices. We have almost completed this experiment, with nearly all the data collected at this point.

In this experiment, subjects first acquire information about a small gasoline engine and then perform two tests of what they have learned. One manipulation–and the only one between subjects–concerns how subjects acquire information about the engine. Subjects acquire information about the engine in one of the following five ways:

- SYSTEMS ONLY: the tutor component of MECH presents the 19 system descriptions for the engine in a controlled, top-down manner
- TERMINAL COMPONENTS ONLY: the tutor component of MECH presents all descriptions for terminal components from the engine in a controlled manner
- SYSTEMS AND COMPONENTS: the tutor component of MECH presents all system and terminal component descriptions for the engine in a controlled, top- down manner
- SUBJECT CONTROL: subjects control the tutor component of MECH to acquire whatever information they wish about the engine
- BOOK: subjects read the sections of the book from which all the information in the tutor component of MECH was drawn

Of general interest is whether MECH promotes better learning than book learning and whether subject control is better than experimenter control. Of more specific interest are the relative merits of the different modes for acquiring specific types of engine information. Which modes lead to the best learning of components? Which lead to the best learning of internal relations? Which lead to the best learning of external relations? Which lead to the deepest learning, hierarchically speaking (i.e., which promote learning of the most specific subsystems)?

After subjects acquire information about the engine, they are tested first for cued recall and then for recognition. In cued recall, a subject receives the name each subsystem, one at time, and has to describe its structure and function. In recognition, subjects perform a timed true-false test on statements about components, internal relations, and external relations from all 19 subsystems.

We have developed a coding system for the cued recall test that specifies the components, internal relations, and external relations that could be recalled for each subsystem. We also have coding categories for various types of errors that subjects may make in recalling these types of information. Of interest will be how correct recall and errors vary with learning mode.

2.4.2 Future Experiments

At this point, we have no plans to run any further learning experiments during the remainder of this contract, although we are very interested in extensively studying the learning of mental models at a later time. Instead we plan to focus our modest resources for the coming year on running some initial studies of troubleshooting. In particular, we plan to explore several of the nine paradigms described in Section III.B. At this point, we are not sure exactly which paradigms we will have explore first. As can be seen from that section, however, there is no shortage of things to do. Eventually we would like to explore all of these paradigms.

3 Learning by Understanding Expert Diagnoses

Diagnosis of malfunctions in complex systems requires a great deal of domain specific knowledge, and any diagnostic reasoner must either already have this information or must be capable of learning it. A frequent means of human instruction in many diagnostic domains is the presentation of cases for students to solve, followed by an instructor solving those same cases. This is true in wide ranging areas such as automobile mechanics, medicine, and management strategy.

In human protocol studies of diagnostic behavior, Lancaster and Kolodner [1987, 1988] examined the evolution from novice to expert in auto-mechanics students.

One type of learning observed in these protocols was "Learning By Understanding Explanations" (LBUE) [Redmond and Martin 1988]. In LBUE, a student, who has only an incomplete understanding of auto diagnosis and repair, follows along as an instructor explains how to diagnose a problem in a particular car, making inferences from the instructor's explanations and actions. When the instructor explains something that deviates from the learner's understanding, the learner debugs his model, adding or changing information. In addition, the learner generates diagnostic short cuts, called symptom fault sets, which directly associate a given fault with possible hypotheses.

In this paper, we present a computational model of LBUE that is implemented in two computer programs EDSEL1 and EDSEL2 (Explanation and Diagnosis: The use of Symptoms, hypotheses, and Explanations for Learning). The novel characteristic of this approach is its ability to use new knowledge that is provided by an instructor to fill gaps and debug an incomplete and incorrect causal domain model. LBUE begins to explain how two different students, with differing initial knowledge, might learn different information from the same example.

The paper will present the general concept, discuss how our approach deals with issues in completing and correcting causal models, and present the two preliminary implementations.

3.1 LBUE - General Process

3.1.1 Introduction

Learning by Understanding Explanations (LBUE) assumes that if a reasoner can explain a teacher's actions, then she can more appropriately generalize what is learned. The explanation will help identify novel situations in which the action is also appropriate. This is essentially the foundation of explanation based learning (EBL) [DeJong, 1983; DeJong & Mooney, 1986; Mitchell et al., 1986]. However, LBUE further acknowledges that the reasoner may not yet know enough to provide a complete explanation. In such a case, the teacher might provide a partial explanation that could assist the student in forming a complete account of the action. If a complete explanation can be formed, then the conditions under which the action is appropriate can be generalized. Furthermore, the student simply can learn the information contained in the partial explanation with the hope that the new information will help explain the instructor's actions in future situations.

For example, an instructor in diagnosis tells students the hypotheses that he generates given a certain problem state. The students must then explain why a particular hypothesis is appropriate and what about the problem state makes it appropriate.

If the student can account for such hypotheses, then that student will be better able to generate appropriate hypotheses in the future. If, on the other hand, the student is too much of a novice to understand a given hypothesis, then the instructor can provide a partial explanation that will direct the novice to a more complete understanding. Not only will the novice be able to generate appropriate hypotheses in the future, but she will also be better able to explain future hypotheses.

The LBUE technique generalizes beyond diagnosis. A mathematics instructor who is teaching algebra tells students what transformations are appropriate given a particular series of symbols. Again, it is the students' job to explain why a particular transformation is appropriate and what it is about the series of symbols that makes the transformation appropriate. Just as in the diagnosis example, the instructor can provide novices with the knowledge necessary to explain the application of transformations. So again, the novice will not only learn what transformations to perform and when, but will also be better able to explain why particular transformations are performed.

The general LBUE process, which follows, allows the student to improve his performance and to improve his explanation of an instructor's actions.

1. *Instructor* states and refines the problem's initial state and goal.
2. *Student* attempts to generate an action.
3. *Instructor* generates an appropriate action.
4. *Student* attempts to explain the instructor's action.
5. *Instructor* provides a partial explanation of the action.
6. *Student* adds the partial explanation to their knowledge and uses it to attempt complete explanation of the instructor's action.
7. If the student generates an explanation in steps 4 or 6, then the explanation will be used to generalize the conditions under which the action is appropriate.

The student, in general, requires a background domain model that is not necessarily complete or consistent and a set of procedural or declarative rules that direct the search for explanations in the current domain. An example of these rules include:

- A hypothesis is explained by causally connecting it to an observed symptom.
- A transformation is explained by causally connecting it to either a major or minor goal of an algebra problem.

3.1.2 Diagnosis as LBUE

The EDSEL LBUE research specifically explores a student's explanation of an instructor's hypotheses and subsequent explanations in a diagnostic domain. When the student is attending to an expert's diagnosis, she must make inferences from the given information to fill in the omitted information. In general, the instructor cannot explain everything at every level of detail. One way to infer the missing information is to attempt *causal chaining*. When the student is told a symptom, she may construct backward causal chains to all possible findings that could lead to that symptom. When she receives a hypothesis, she might build forward chains to all possible findings that could be caused by the hypothesized fault. If forward and backward chains meet, then the student has an explanation for the hypothesis and thereby has filled in the omitted information. The student might finally collapse the chain into a fault and symptom pair and save that as a symptom fault set. The collapsed chain could then be used more efficiently in future, similar situations.

Often a student will not have enough information to explain the hypothesis. In this case, the instructor's explanation can be useful. First, if the explanation represents a new causal relation, such as X causes Y, it can be added directly to the causal domain model with high credibility. Second, it may allow bridging a gap to complete a *causal chain*, thus enabling the student to add a collapsed *causal chain* as a new association in the causal knowledge, as in EBG. Third, when the explanation does not complete a chain, the student may still infer that the hypothesized fault causes X, and that Y causes the symptom, though there may be unknown intermediate causal links.

The LBUE concept goes beyond explanation-based methods by being able to use new information to complete a *causal chain* that would not be possible otherwise. As has been noted by many researchers [Mitchell et al., 1986], EB methods do not generate any knowledge that the student or automated reasoner does not already have; rather, existing knowledge is reorganized to be more useful. In collapsing the chains, the LBUE method has some similarities to Explanation-Based Generalization (EBG) [Mitchell et al., 1986] and Explanation-Based Learning (EBL) [DeJong & Mooney 1986]. The student has used available knowledge to form an explanation, which is then stored for later use, somewhat like a macro operator in STRIPS [Fikes et al., 1972]. The LBUE concept goes beyond explanation-based methods by allowing learning of truly new information and allowing collapsing of a *causal chain* when the domain theory is incomplete.

3.1.3 An Example

A example will illustrate the ideas and introduce the LBUE algorithm for diagnosis. An instructor may present the student with a malfunctioning car in which the engine cranks but does not start. She may suggest a hypothesis that the distributor cap is cracked. The student may know, from whatever sources, that for a car to start, the starter motor must turn, and combustion must occur. The student may further know that for combustion to occur, there must be fuel mixed with air in the cylinder, and a spark from the spark plug. Therefore, the student might reason backwards from the symptom:

```
(not (start engine)) is-caused-by
   (not (combustion cylinder))  is-caused-by
      (not (ignite spark-plug))
```

However, he might not know what a cracked distributor cap can cause, so he does not understand why that fault was hypothesized. In other words, he might not be able to reason from (CRACKED DISTRIBUTOR-CAP) to what that condition causes.

The instructor might then provide a partial explanation that a cracked distributor cap can allow moisture to collect inside the distributor. The student did not previously know this causal relationship, and the relationship could not have been generated without some outside information. The student, using the new information, might finally be able to completely explain the hypothesis:

```
(cracked distributor-cap)      causes
   (contains distributor-cap moisture)    causes
      (low (input spark-plug electricity)    causes
         (not (ignite spark-plug))             causes
            (not (combustion cylinder))            causes
               (not (start engine))
```

A completed explanation of this form will be termed a *causal chain*. With the complete causal chain, the student understands the hypothesis (and the explanation). He can learn that a cracked distributor cap causes the symptom of the engine cranking but not starting. If the student was missing the fact that moisture in the distributor cap can cause less electricity to reach the spark plug, he may still be able to infer that fact based on the explanation having been given by the trusted expert in conjunction with general knowledge about water's effect on electricity. In summary, the partial explanation permits complete *causal chaining*, it assists the student in directing the search for relevant causal relationships, and it can be learned to improve future explanations of hypotheses.

3.1.4 Algorithm

LBUE, as applied to learning automobile diagnosis from examples, requires a very straightforward algorithm as illustrated in the above example. The algorithm must process symptoms, hypotheses, and explanations, and must add to the causal domain model when possible. An outline of this algorithm as it may be carried out by humans or artificial reasoners follows.

1. From the symptom, chain backward toward possible findings.
2. From each hypothesis, chain forward toward possible effects.
3. If the symptom chain meets a hypothesis chain, then the generalization that *(cause hypothesis symptom)* is added to the symptom fault table and to the causal domain model.
4. If the chains do not meet,
   (a) Chain backwards from the explanation toward the hypotheses.
   (b) Chain forward from the explanation toward the symptom.
   (c) If both directions can be linked, then the most general relationship *(cause hypothesis symptom)* can be learned.
5. Add explanation to the causal domain model.

This algorithm is a specialization of the *student* portion of the general LBUE process (section 2.1) for diagnosis.

The inputs to the algorithm are a diagnostic example and an initial causal domain model (see section 3). The output is an updated causal domain model and a cognitive trace of the learning process. The following knowledge may be learned:

1. new objects,
2. structural relationships between objects,
3. new causal information, and
4. new symptom fault knowledge.

After this learning, the causal domain model is more capable of verifying an explanation, and diagnosis is more efficient and more powerful for the same or similar problems.

3.1.5 Example 2

The algorithm is demonstrated in two further examples. For clarity, all examples present a single path through the causal model, rather than the tree-like search that is more typical. The current example diagnosis demonstrates learning with just the symptom and hypothesis, without requiring an explanation. The symptom that the instructor reports is slow cranking of the engine. The student chains backward hypothesizing that the crankshaft is spinning slowly, that the starter motor is spinning slowly, that the battery is not generating much electricity:

```
(slow (crank engine)) -->    ; meaning of the engine cranking
   (slow (spin crankshaft)) -->            ; cause
      (slow (spin starter-gear)) -->       ; cause
         (slow (spin starter-motor)) -->   ; cause
            (low (contains starter-wire current)) --> ; cause
               (low (generate battery electricity))
```

The student achieves this backward chaining by using information about normal function and how, in general, modification of that normal function modifies the rest of the mechanism.

The instructor then suggests the hypothesis that a cracked wire can cause the observed symptom of slow cranking of the engine. The student, if applying the algorithm, chains forward, inferring that the battery cable may be carrying less than normal current, that the starter receives less power, and that the wire in the starter receives less electricity:

```
(cracked battery-cable) -->
   (low (contains battery-cable current)) --> ; cause
      (low (input starter current)) -->       ; cause
         (low (contains starter-wire current))
```

In this example, the symptom and hypothesis chains meet, so an explanation is not required. The student would then understand the instructor's hypothesis, and is able to learn a new symptom-fault set and add the relationship

```
(cause (cracked battery-cable) (slow (crank engine)))
```

to the causal domain model.

3.1.6 Example 3

As in the first example, the student does not always have enough domain knowledge to fully understand an instructor's hypothesis. When an explanation is given, an important gap in the student's knowledge may be filled, thereby allowing completion of a *causal chain*. Additionally, the explanation may assist the student in directing the search for relevant causal relationships. For example, there are potentially many *causal chains* that might be constructed when the symptom is that the car is stalling. The explanation can help determine which of many chains is relevant.

Suppose that the instructor presents a situation in which the car has stalled. The student chains backward, hypothesizing only that there might not be combustion occurring in the cylinders:

```
(not (run engine)) -->        ; meaning of a stalled engine
   (not (spin crankshaft)) -->            ; cause
      (not (down-stroke cylinder)) -->
         (not (combustion cylinder))      ; any cylinder.
```

The instructor then provides the hypothesis that the butterfly valve of the choke assembly is stuck. Forward chaining would reveal:

```
(not (movable butterfly-valve)) -->
   (low (flow air carburetor))
```

In this case, the forward and backward chains do not meet, and the student does not know or cannot retrieve any causal relationships that might connect the chains. However, if the instructor provides the explanation that low air flow into the carburetor leads to a low air/gas mixture as the air passes the fuel float bowl then the following results:

```
(not (movable butterfly-valve)) -->
   (low (flow air carburetor))     -->
      (low (mix air gas))          -->
         (not (combustion cylinder))     -->
            (not (down-stroke cylinder)) -->
               (not (spin crankshaft))   -->
                  (not (run engine))
```

This example of explanation demonstrates how a new causal relationship may be added, (low (flow air carburetor)) $\Rightarrow$ (low (mix air gas)) , and how an existing causal relationship that was not accessed or was not known to apply, (low (mix air gas)) $\Rightarrow$ (not (combustion cylinder)), can be used. The student, therefore, understands the instructor's hypothesis, and is able to learn a new symptom-fault set and add the relationships

```
(cause (low (flow air carburetor)) (low (mix air gas)))
(cause (not (movable butterfly-valve)) (not (run engine)))
```

to the causal domain model. The following section will be more explicit about what is contained in the causal domain model.

3.2 Causal Domain Models

Causal domain models describe some complex system to allow a reasoner to perform tasks such as diagnosis, prediction or simulation. They are usually used to represent physical devices, but can in general represent any system including biological and social systems. Causal domain models may represent only the normal behavior of the system, or may represent both normal and faulty behavior. The benefit of reasoning from causal domain models is that more knowledge is available to the reasoner than just simple associations. Circumstances or problems that had not been anticipated can be handled by making inferences from the normal function. This generates a more robust, less brittle problem solver. Chandrasekaran and Mittal's [1982] MDX demonstrated this by generating rules from a model of normal function to provide a more complete understanding of the modeled system. Johnson-Laird [1983] has argued that people achieve their high level of performance by building "mental models", rather than just reasoning from propositions. A simple causal domain model would correspond to his concept of relational models, a more complex model would correspond to his temporal model concept.

In order to support the type of reasoning desired and displayed by humans, a causal domain model must represent several different types of information. A component's inputs and outputs should be represented. This allows the reasoner to trace inputs and outputs until a problem is found. If component x does not have the proper output, then if its inputs are correct it is not working correctly. If it receives bad input then the component that is supposed to produce that input as its output is suspect. In a simulation task, representing inputs and outputs allows reasoning about the effects of changing something in the system.

A closely related type of knowledge is that of connections between components. This allows tracing effects to components that might not seem to be related based on inputs and outputs, but which are adjacent. For example, the adjacency between coolant and an engine causes the engine to cool down. Abbott [1988] gives the example of the blade of a fan breaking off and damaging nearby components.

A second type of knowledge necessary in a causal domain model is information that represents a partonomic hierarchy. Any given component is part of a more complex component, and is made up of less complex parts. Sometimes it is better to reason at more abstract levels, such as the fuel system causes fuel to get from the gas tank to the carburetor, or that the charging system is not working. Other times it is more useful to reason at a more detailed level, such as that the starter motor brushes cause a reverse in polarity, causing the motor to spin the pinion gear, or that the carburetor discharge check valve is stuck. For example, Lancaster and Kolodner [1987] observed that the advanced students and experts tended to carry out diagnosis in a hierarchical manner, isolating the faults first to an abstract level, then refining it to a more detailed level. This could also be applicable in simulation. The higher level simulation can be done first, then more detailed simulation can be done if time and resources permit. Or the most critical part could be refined while other parts are left less specific.

A third type of necessary knowledge is an arrangement of components into a specialization or isa hierarchy. The reasons for this are similar to those for having a partonomic hierarchy. Such a hierarchy allows reasoning at the appropriate level of abstraction. In this case, it enables reasoning about belts, gears, hoses, or wires when appropriate, instead of always at the level of a specific component.

Lastly, and probably most importantly, there must be a representation of causality. This describes the function of the system and at least indirectly the way the system malfunctions. A reasoner is able to test the reasonableness of a hypothesis by explaining or simulating its probable effects. As mentioned above, the causal knowledge used to explain or simulate can either reflect normal or faulty functioning. In addition to being important in its own right, causal knowledge also enables learning, as we discuss in the context of LBUE.

The knowledge necessary in a causal domain model can be shown with examples from EDSEL1 and EDSEL2. The implementations represent components of the car in a manner roughly equivalent to frames [Minsky 1975] or objects [Goldstein 1980]. Information about any system component includes all of the above types of knowledge. For example, Figure 3 shows a representation of a starter, and Figure 4 shows a more detailed representation of a carburetor barrel from EDSEL2.

As many AI researchers have discovered, complete and consistent causal domain models are very difficult to design and maintain. In fact, even the human designer of such models often does not have an error free understanding of the system to be modeled. Fortunately, people, including AI researchers, are able to function successfully with clearly incomplete, inconsistent models of the world. For example, many people believe the intuitive, but long since disproved, impetus theory of physics. An important goal for AI research is to allow an artificial reasoner to also function successfully with incomplete, inconsistent models. AI causal domain models will be increasingly incomplete and inconsistent as the complexity of the modeled system grows. For example, in EDSEL2 an admittedly incomplete representation of a carburetor involves 25 frames. In creating large models of entire systems, human working memory limitations prevent noticing most contradictions, since all assertions cannot be compared with all related assertions. We will next discuss how a human or automatic reasoner can recognize what knowledge is missing from the causal domain model, and how the reasoner can correct the deficiencies using the LBUE approach.

3.3 Issues for Completing and Correcting Causal Models

Two major problems arise because complete and correct causal models for non-trivial systems are, in general, unattainable. First, the reasoner must use the domain model while it is incomplete and incorrect to generate acceptable diagnoses. Second, the reasoner must augment and correct the model. The LBUE approach addresses both of these problems. Furthermore, LBUE is not sidetracked by coincidences the way similarity based learning approaches are, and the method does not attempt to extend the explanation of the diagnosis to the level of quantum mechanics. In this section we will discuss how LBUE addresses the problem of augmenting and correcting the causal model.

As discussed above in Section 2, a good diagnostic reasoner tries to explain why an hypothesis causes a symptom. It is, in fact, this process that allows for the recognition of different types of missing information, and that mediates the addition of knowledge to the causal model. The process of explaining hypotheses identifies just that information that might be useful for diagnosis because diagnosis is itself explanation, and hence requires the same information. That is, performing a diagnosis requires knowledge of causal relationships that lead from the symptom to the fault, which are the same relationships required to explain the instructor's hypotheses. The following discusses the types of information that can be missing from the causal domain model, how that information can be detected and added , and how inconsistent knowledge is handled. This section elaborates on a discussion of these issues in Martin and Redmond [1988].

3.3.1 Types of Missing Knowledge

The causal model can be missing several types of knowledge. These include some of those mentioned in Section 2.4:

- Causal relationships
- Other relationships between objects
- Objects.

This also includes nonoptimal organization of the knowledge that would lead to inefficiency in diagnosis. We do not discuss symptom-fault associations here, because we currently consider them as separate from the model.

Causal Relationships

The most important type of missing knowledge is 'causal relations'. In general, a causal model may be missing many causal relations that are necessary for diagnosis. A reasoner will recognize that they are missing a causal relationship when an explanation of a symptom cannot be formed, either while watching an instructor or while actually doing diagnosis. As well, there are situations in which an unknown causal relationship will be presented to the reasoner. Both possibilities are simple to detect, the former when *causal chaining* fails or when no reasonable hypothesis is generated, and the latter, when the reasoner is actually told that something is missing. For instance, in Example 3 (section 2.6) above, when the backward chaining from the symptom

```
(not (run engine))
```

and the forward chaining from the hypothesis

```
(not (movable butterfly-valve))
```

have been attempted, the reasoner knows that he is missing knowledge, since he cannot explain why the instructor would make that hypothesis. Also, when the explanation,

```
(cause (cracked distributor-cap)
       (contains distributor-cap moisture)),
```

contains knowledge that the causal domain model does not contain, the reasoner only needs to check the model to realize that the information is missing.

Other Relationships

Another form of missing knowledge that is easily detected are 'referred-to facts'. These facts can involve various non-causal relationships. Specifically, when an object or general relationship between objects is asserted, but is not known, then it is missing from the model. Somewhat more interesting, though, are when such relationships are merely implied. The reasoner guesses he does not know an implied fact when a causal relationship is stated that the reasoner believes requires a mediating fact. For example, a reasoner may know,

```
(INTERLOCKED gear1 gear2) & (SPIN gear1 'clockwise)
                                --> (SPIN gear2 'c-clockwise)
```

and an instructor may state,

```
   (SPIN starter-gear 'clockwise)
                --> (SPIN flywheel-ring-gear 'c-clockwise)
```

From this, the reasoner will recognize that he may be missing a fact (i.e., that the two gears are interlocked).

Objects

The domain model must add a necessary object when a hypothesis, symptom or explanation refers to the object and the reasoner cannot retrieve any information about the object when he tries to form an explanation. As long as the reasoner does not need to reason about the object he can remain blissfully ignorant of its existence, thus avoiding the need to understand atoms and molecules.

Inefficiency

In a sense, inefficiently represented knowledge is a type of incomplete knowledge. The information that is needed is in the causal model, but is not useful because either it cannot be accessed, it is given insufficient credibility, or it leads to slow processing. We refer to this type of missing information as efficiency information. The reasoner must be able to arrive at a reasonable or correct hypothesis quickly. If he cannot, the causal model must be modified to ensure timely and correct diagnoses in the future. The reasoner can recognize that he is missing this kind of information if he arrives at an incorrect hypothesis during diagnosis or is slow at generating a correct hypothesis.

3.3.2 Methods of Handling Incomplete Knowledge

The LBUE approach handles each of the above types of missing knowledge. We will discuss the methods for adding missing:

- Causal relationships
- Other relationships between objects or facts
- Objects
- Efficiency information

Causal Relationships

An instructor's explanation of a given hypothesis can lead to information being added in three different ways:

- The explanation is an unknown causal relationship.
- The explanation can enable filling a gap in a *causal chain.*
- The lack of a complete explanation can trigger an inference that enables filling a gap in a *causal chain.*

An example of the explanation itself being an unknown causal relationship which can be added to the model directly follows. If the instructor explained that,

```
(cause (corroded battery-terminals)
       (not (connect battery battery-terminals)))
```

and this relationship was not associated with either battery or battery-terminals in the causal model, then it can be added there. The second way that the instructor's explanation can be used is to enable filling a gap in a *causal chain.* Either the explanation filled the gap, or it was a better cue to information that was not being accessed in the causal model. If, as in Example 3 (section 2.6) above, the *causal chain* that can be built from the symptom (NOT (RUN ENGINE)) is:

```
(not (run engine)) -->
   (not (spin crankshaft)) -->
      (not (down-stroke cylinder)) -->
         (not (combustion cylinder))
```

and the *causal chain* that can be built from the associated hypothesis (NOT (MOVABLE BUTTERFLY-VALVE)) is:

```
(not (movable butterfly-valve)) -->
                 (flow air carburetor low)
```

then there is a gap in the *causal chain* — the hypothesis is not fully explained. If the instructor provides the explanation that low air flow into the carburetor leads to a low air/gas mixture as the air passes the fuel float bowl then the following results:

```
(not (movable butterfly-valve)) -->
   (flow air carburetor low) -->
      (mix air gas less)  -->
         (not (combustion cylinder)) -->
            (not (down-stroke cylinder)) -->
               (not (spin crankshaft)) -->
                  (not (run engine))
```

Not only is the causal relationship from the explanation used in filling the gap, but the relationship that (MIX AIR GAS LESS) causes (NOT (COMBUSTION CYLINDER)) is accessible under carburetor when it had not previously been available. Between these two processes, the *causal gap* is filled, and a new causal relationship can be added to the model.

The third way in which the instructor's explanation can be used is to infer a relationship that would fill a *causal gap*. If the explanation is not sufficient to fill the gap as discussed above, causal relationships, which are implied by the expert instructor, can be inferred. The instructor implies that there is a *causal chain* between the hypothesis and symptom and that the explanation lies along this *causal chain*. In general, gaps will be filled with inferences if there is some knowledge that indicates that a cause is plausible. For example, a cracked wire can cause low electricity because (a) wires conduct electricity, and (b) a conduit affects what it conducts. An inferred cause is given lower credibility than other learned relationships. In situations where there are several plausible but mutually inconsistent inferences that might fill a gap, the chosen causal relation could depend on confirmation from the instructor.

Other Relationships

Knowledge can also be added to an incomplete causal model by inferring facts from a cause. This would occur as a result of the starter-gear/ring-gear example mentioned above. In that case, the reasoner will infer that the starter gear and ring gear are interlocked.

Objects

A newly discovered object cannot be fully specified when it is added to the model, since the object is only indirectly taught. The object, though, can be included in the model along with any knowledge explicitly given, such as a causal relationship given in the explanation involving the object. It is conceivable that the object could be determined to be an instance of something more specific in the isa-hierarchy than 'object', thus enabling more class information to be inherited, but this is not currently done in either implementation.

Efficiency Information.

The final type of missing information that a reasoner must handle is efficiency information. For instance, the explanation can allow the access of knowledge that could not previously be accessed. Additionally, filling a gap in a *causal chain*, as discussed above, is a way of dealing with some inefficient knowledge. This allows collapsing the chain into a single causal relationship, which can be used for more efficient processing. In order to allow for proper generalization of variables when collapsing a *causal chain* [DeJong & Mooney, 1986], a substitution list is kept that indicates to what categories each feature in the example was matched when instantiating the causal relationships. The collapsed chain then uses the most general category that describes a particular feature as the variable name in the antecedent or consequent of the new causal relationship.

3.3.3 Inconsistent Knowledge

Since new information is being added to the causal domain model there is a possibility that a contradiction may occur. This raises three issues that we will discuss:

- Types of inconsistencies
- Detection of inconsistencies
- Handling of inconsistencies.

Types of Inconsistencies

A contradiction could arise in many different ways. There might be two pieces of knowledge such that

```
     (corroded battery-terminals) -->
                              (connect battery battery-terminals)
 and (corroded battery-terminals) -->
                        (not (connect battery battery-terminals))
```

A circular chain might be possible from known information, such that a condition indirectly causes a contradiction of the condition:

```
   (corroded battery-terminals)
   --> (not (connect battery battery-terminals))
   --> (not (spin starter-motor))
   --> (not (spin starter-gear))
   --> (not (corroded battery-terminals))
```

or such that a condition indirectly causes itself.

Detection of Inconsistencies

Contradictions may not be detected immediately, though, because the causal information is distributed throughout the causal model, and because an arbitrary amount of *causal chaining* might be necessary to detect the contradiction. A causal relation indexed under one car component in the model might contradict a causal relation indexed under another component. Without an exhaustive search of the model before or after each addition to the model, it is impossible to guarantee that the model will be completely consistent. Even more seriously, since the inconsistency may not be detected without exploring all possible causal inference chains derivable from the model, checking for inconsistency is intractable. One heuristic method for correcting some inconsistencies is to use the problem solving task to drive inconsistency detection. When the reasoner constructs a *causal chain*, she checks the chain for contradictions. This is limited to direct contradictions, with no inferencing done beyond the already built *causal chain*. This method limits the amount of work done.

Handling of Inconsistencies

In cases where the new input is found to be inconsistent, the source of the information is the first information used to decide what to believe. Knowledge from the expert is given precedence over inferred, learned, or background model information. Information that contradicts the expert can either be removed or its strength can be decreased. When both pieces of information were inferred, or one was inferred and the other is background information, the piece with higher plausibility value or strength is more highly believed. As with expert information, the lesser believed information is removed or its strength is decreased.

3.4 Implementations

Implementation of the general model described above has followed two parallel paths, EDSEL1 and EDSEL2 (Explanation and Diagnosis: The use of Symptoms, hypotheses, and Explanations for Learning). This decision was made because the research is exploratory and, therefore should generate several alternative approaches to the problem. This allowed parallel exploration of the problem space, and did not require premature agreement on one approach. Both implementations are models of the auto-mechanics student evolving to greater expertise. EDSEL1 uses a fine-grained network representation of the causal model, while EDSEL2 uses a more traditional frame or object based causal model. These alternatives can be productively compared. An additional advantage of the separate implementations is that the different programming efforts highlight different inconsistencies and difficulties with the model, possibly producing a more general theory than would arise from a single implementation. Descriptions of each of the implementations follow.

3.4.1 EDSEL1

One of the implementations, EDSEL1, is based upon a simple active semantic net, similar in process to local connectionist models such as McClelland and Rumelhart's [1981] interactive activation model. This approach was taken to examine how parallel competitive activation might be applied in LBUE diagnosis.

LBUE Process

In a diagnosis training example, EDSEL1 receives input, consisting of symptoms, hypotheses, and explanations of those hypotheses, as described above in the general algorithm section (section 2.4). The symptom will trigger backward chaining through the existing causal knowledge where the chaining is guided by weights on the causal links that indicate credibility or likelihood. The underlying causal knowledge was designed with reference to Kuipers [1984] and DeKleer and Brown [1981]. To process the symptom, the system attempts to discover the most likely cause.

As the instructor progresses in the problem, she will suggest possible hypotheses. For each, the system will try to explain why that hypothesis would cause the symptom, again the search is guided by weights on the causal links. If a *causal chain* can be found from the hypothesis to the symptom then a "short cut" cause is added, of the form *(cause hypothesis symptom)*. The antecedent of the collapsed cause would also include any preconditions of causes in the *causal chain*, *(cause (pre1 & pre2 ... & hypothesis) symptom)*. This essentially establishes the causal context in which the hypothesis causes the symptom. The procedure is simple until one considers variables in the antecedents and consequents. In order to allow for proper generalization of variables [see DeJong & Mooney, 1986], a substitution list is kept that indicates to what slots or frames each feature in the example was matched in order to instantiate each causal relationship. The collapsed chain then uses the most general frame that consistently refers to a given feature as the frame name in the antecedent or consequent of the new causal relationship. For instance,

```
      (drop ?object) --> (fall ?object)
      (fall ?object) --> (forceful-contact ?object ?surface)
      (forceful-contact ?rubber-object ?surface) --> (bounce ?rubber-object)
```

applied to '(drop ball)' would result in,

```
      (drop ?rubber-object) --> (bounce ?rubber-object)
```

where '?object' and '?rubber-object' are abstractions that serve as variables. The algorithm in this implementation for collapsing chains differs from the general algorithm in that the frame substitution list is applied while collapsing.

When explanations enter the system, they are assumed to indicate what *causal gap* filling is required. They may simply fill a gap themselves, or they may refer to causal knowledge that the system already has as indicated in section 4.2.1. The explanation is handled by continuing to process the previous hypothesis, emphasizing those chains that contain the explanation. Failing that, the system attempts to fill *causal gaps* from the end of all hypothesis chains to the provided explanation and from that explanation to the start of all backward chains from the symptom.

If the system is still unable to find a *causal chain* even after given an explanation, it attempts to bridge a gap between two partial chains. This is the situation depicted in section 4.2.1. When no chain is found, several possible incomplete paths may have been considered. If the end of an incomplete chain can be connected to the beginning of a chain from the symptom, then that *causal gap* will be filled and the entire chain collapsed as above. *Causal gaps*, however, will be filled only if there is an explanation or some general knowledge that indicates that a cause is possible. An example of this possibility is that, a cracked wire might be assumed to cause low electricity because (a) wires conduct electricity, and (b) a conduit affects what it conducts. This is a reasonable guess given some general knowledge about what causes what.

Whenever erroneous causes are encountered, they are given lower strengths or weights, but retained to be used if all other alternatives fail to be useful for diagnosis. An incorrect cause as discussed in section 4.3 might have the form (cause A B) where the instructor states or implies that (cause A C) and the system knows that B and C are incompatible states — states that cannot coexist. The behavior of this subsystem would be improved if the system attempted to explain the contradiction in order to isolate its necessary and sufficient features.

The system currently learns and partially debugs information necessary for diagnostic performance. It acquires new causal information, collapses chains of reasoning for more efficient use, and reduces the weight of causes that contradict what the instructor states or implies is a true causal relationship.

Representation and Basic Processes

The representation used in EDSEL1 is essentially a network of parallel computing elements connected by weighted links. This is currently implemented on a sequential machine using what amount to weighted rules to handle the links. The connectionist idealization was adopted because it provides an efficient algorithm for competition between weighted rules and because it has potential as a human model. This metaphor when combined with distributed representations may also allow a future version of EDSEL1 to take advantage of similarities between rules.

Figure 5 demonstrates the network organization of EDSEL1 and shows how the network can support the frame/slot/value notion. In particular, the figure depicts the knowledge that a battery is connected to a starter motor by a wire that conducts electricity. It uses an abstraction about conduits that connect objects to organize the information. In this case, 'connect001' is a frame that inherits slots from connect. 'Input', 'output', and 'conduit' are all slot names, and 'material' and 'flow-obj' are slot names for the frame, 'conduit'. Additional token nodes, such as 'input001' are specialized slot names that are required for unambiguous interpretation of the network. In general, interpretation of this knowledge is performed by simple competitive spreading activation from the slot name and the frame name to the value or values. For instance, the value for the 'conduit-material' of the connection from the battery is 'electrical wire'. As this example demonstrates, slot names, such as 'input', can be more general than the token nodes, such as 'input001'. This is possible because the system knows the abstract category to which each token belongs.

Causal information is represented in the same manner, using a 'cause' slot and the antecedent as the frame. Figure 6 presents an example of this general method for representing causation. It depicts the fact that dropping an object makes it fall which, in turn, causes it to contact the ground. When there are multiple antecedents for a particular consequent or multiple consequents for a particular antecedent, the mechanism places the multiple values in competition. In particular, all links in the representation have associated weights that determine the strength of each of the possible results. Each of the competing values also inhibits the other's strengths in proportion to their own strength. The result is that one of the possible values will have a very high strength and that value will be used for later processing. Figure 7 depicts a symptom — the car cranks, but does not start — that could be caused by one of two possible faults. Either the distributor cap contains water or the fuel line contains dirt. Suppose that the weight on the link between 'contain001' — the first hypothesis — and the symptom is 0.7; and that the weight on the link between 'clog001' — the second hypothesis — and the symptom is 0.5. When the symptom, 'cranks, but does not start', is active, it sends activation along its links in proportion to those weights. Clearly, 'contain001' will receive the highest activation and then will further inhibit 'clog001'. The frame names that are causally related to the symptom receive higher activations than do other related frames because the cause slots are activated by the request for causal information. Therefore, the best answer to a 'cause' request will be a value or frame associated to a cause unless none exists. What is the point, one might ask, of having multiple possible answers to a request if the same one is always chosen? In fact, the multiple possible results can all be generated by the mechanism, depending on what additional facts happen to be true in a given situation. For example, if the system knows that a lot of water was introduced to the engine, 'contain001' would easily receive the highest activation. However, if the system knew that dirt had been introduced to the fuel tank, then the second hypothesis, 'clog001', might edge out the other hypothesis by virtue of additional activation from the 'clog' slot.

Example

Figure 8 displays a small amount of causal information that a mechanic might have about a battery and electricity. The figure includes knowledge about leaks, but not about what a leak can cause. It also has information about what happens when the ignition switch is turned on, the function of a battery, and what flows between the battery and the starter motor. If an instructor states that he believes that a hole in a battery may be causing slow cranking of the engine system and that holes cause leaking, then Figure 9 is the resulting memory structure. Note that the system learned that holes cause leaking, battery holes cause electrolytes to leak out, and that a battery hole causes slow cranking of the engine system. The reader should note that, for simplicity, the figures only represent small pieces of the resulting knowledge and that they omit the connections to the rest of the causal domain model. As well, both models omit multiple chains.

In contrast to Figure 8, if another mechanic is given the same information, but begins not knowing that batteries contain electrolytes, s/he will learn that there is some liquid in a battery that is necessary to output full electrical power and that this liquid is called 'electrolyte'. This demonstrates that the existing state of knowledge in the system can help determine what is learned. The system learns what it does not know and it may learn those facts at varying levels of specificity, depending on the inferences that can be made.

In summary, this implementation learns enough to reproduce *causal chains* that it encountered in the training examples. Future directions will include allowing greater learning about the instructor's strategies from the explanations, and allowing learning of the abstract knowledge necessary to evaluate *causal gap* filling in the absence of an explanation.

3.4.2 EDSEL2

Representation

The second implementation, EDSEL2, uses an enhanced version of the causal domain model described in Allison [1987].

The representation uses structure like frames [Minsky 1975] or objects [Goldstein 1980]. It uses basic frame system functions developed by Tom Hinrichs for JULIA [1988]. The current form of the causal domain model contains essentially 5 different types of frames. There are frames for

1. objects, including car components,
2. predicates,
3. modifiers,
4. features, and
5. relations.

The largest percentage of the causal domain model is frames for objects. Every system, component, and part should be represented by a frame. This is certainly not the case at this point, only the starter and the carburetor have been represented in detail with multiple levels of abstraction. Other parts of the car have been represented at only one level, mainly major parts such as battery, air filter, alternator, and battery cables. Car components are frames with slots for

- parts - the objects that comprise the given object
- part-of - the higher level entity that the object is part of
- input - objects that the object needs to function, and their source
- output - entities that the object produces, and where they go
- connected-to - objects that this object is connected to

- causal relations - what the object is supposed to do and how it fails, represented in terms of relationships between components.

This representation constructs a partonomic hierarchy, as well as representing the normal functioning of the car and creating a isa hierarchy. So far no effort has been expended on dealing with different cars having different parts. At this time, the model has on the order of 125 objects.

All predicates such as CORRODED, CONNECTED, CRANK, CONDUCT, and HOLE, and all modifiers such as levels – HIGH, LOW; speeds – FAST, SLOW; sizes – LARGE, SMALL; quantities – MORE, LESS; and derivatives – INCREASE, DECREASE are also represented by frames so that they can be interpreted by more general functions. Frames were created for predicates so the system could reason about predicates rather than having knowledge of predicates embedded in the code. For example, a predicate such as connected must be handled differently than a predicate such as corroded. If some state or action causes something to be corroded, then that is knowledge about problems rather than normal function. Also, some predicates refer to one object (e.g. corroded) while others refer to two objects (e.g. connected). A nice frill is the specification of the type of modifier the predicate takes. The specification of the type of modifier for the predicates relies on the definition of modifiers. Modifiers are predicates that modify other predicates, such as speed, size, quantity, etc. If something causes an object to have a hole, then a greater magnitude of that something will cause (large (hole ...)). If something causes an object to be clogged, then a greater amount will cause it to be (more (clogged ...)). The predicate 'hole' is modified by a size, while 'clogged' is modified by a quantity. These types of modifiers are defined with greater than normal and less than normal values. This allows the system to reason about modifying predicates. For example, as in the above discussion of predicates, it can be important to know if something implies a greater than normal value. Also, based on these frames it can be determined that (slow (crank ...)) is the opposite of (fast (crank ...)).

There are frames for features, that is for slots. For example the car model and the symptom are slots for a problem. There is a frame for 'symptom' that lists the type of data to expect and whether the slot is salient. If salient features can be learned then the frames can be modified; this structure does not necessarily imply an *a priori* assignment of feature salience.

Relations are sort of higher order predicates, a predicate that refers to predicate(s). They are defined somewhat like predicates, since they are more like predicates than modifiers. The main reason these are currently needed is for the number-of-claus slot which helps process clauses involving them by specifying how many places they require. This is not very refined as of yet, 'cause', 'not', 'and', and 'or' are the only defined relations at this time.

Process

The process at first closely follows the general concept presented in section 2.4. We will note correspondences to the steps in that algorithm. The process is pictured in Figure 10. We will use Example 3 (section 2.6) above to demonstrate the steps, and the learning possible.

From the symptom presented as input to the system, the system chains backward toward possible causes (Step 1). In the case of Example 3, the system would chain backwards from

```
    (not (run engine))
```

to

```
    (not (spin crankshaft))
```

to

```
    (not (down-stroke cylinder))
```

and finally get no farther than

```
    (not (combustion cylinder)).
```

Thus (NOT (RUN ENGINE)) would be *S* in Figure 10, and (NOT (COMBUSTION CYLINDER)) would be *Z* in Figure 10.

From the presented hypotheses, the system chains forward toward possible effects (Step 2). In the case of Example 3 (section 2.6), the system would chain forward from

```
    (not (movable butterfly-valve))
```

and not get any further than

```
    (low (flow air carburetor)).
```

Thus (NOT (MOVABLE BUTTERFLY-VALVE)) would be *H* in Figure 10, and (LOW (FLOW AIR CARBURETOR)) would be *A* in Figure 10.

If the chains meet, as in Figure 10(a), then the generalization that the hypothesis causes the symptom is added to the symptom fault table, and to the causal domain model under the components involved in the symptom and hypothesis (Step 3). For example, if the system had been able to chain as far forward as

```
    (not (combustion cylinder)),
```

in Example 3 (section 2.6), then that would have been both *A* and *Z*, and the relationship

```
    (CAUSE (not (movable butterfly-valve)) (not (run engine)))
```

could be learned and stored in the symptom fault table and also stored in the causal relations under butterfly-valve and engine in the causal domain model.

If the chains do not meet, as in Figure 10(b) where nothing in the chain from *H to A* matches anything in the chain from *S to Z*, then an explanation is needed in order to understand. This is the actual case in Example 3 (section 2.6), and the instructor provides the explanation that low air flow into the carburetor leads to a low air/gas mixture as the air passes the fuel float bowl. The explanation given by the instructor is intended to clarify the reasons for making the hypothesis. Therefore, it is information that he considers hardest or least likely to be known by the students. It can be added to the information in the causal domain model, and can frequently aid in bridging the *causal gap* between the chains. This is done by using causal information included in the explanation, and causal information in the causal domain model under the car component(s) referred to in the explanation.

The system tries to chain backwards from the component involved in the explanation toward the forward chain built from the hypothesis (Step 4a). In Example 3 this does not yield anything new, since the explanation points forward and the carburetor had already been cued, so no new relationships were made accessible.

Then the system tries to chain forward from the component involved in the explanation toward the backward chain built from the symptom (Step 4b). In Example 3 this means chaining from

```
    (low (flow air carburetor))
```

to

```
    (low (mix air gas))
```

to

```
    (not (combustion cylinder)),
```

which of course fills the *causal gap*.

This process is illustrated in Figure 10(c), where chains are built from *E to B* in an attempt to meet the chain from *H to A*, and from *E to Y* in an attempt to meet the chain from *S to Z.* In this case, Y = (NOT (COMBUSTION CYLINDER)), and B = E = (LOW (FLOW AIR CARBURETOR)).

If either direction of chaining is successful in closing a gap, then a causal relationship can be learned. This is illustrated in Figure 10(d), where since *A and B* represent the same thing, the chains from *H to A* and from *E to B* meet, and the partial cause *(cause H E)* can be learned. This is one difference from the general algorithm. This could be the case in Example 3, since the gap, in the opposite direction, from E = (LOW (FLOW AIR CARBURETOR)) to Z = (NOT (COMBUSTION CYLINDER)) is filled (since *Y = Z*), the short collapsed chain

```
(CAUSE (low (flow air carburetor)) (not (combustion cylinder))
```

can be learned. However, in this instance more valuable information can be learned.

If both directions can be linked, as in Figure 10(e) where *A and B* represent the same thing and *Y and Z* represent the same thing, then the most general relationship *(cause hypothesis symptom)* (or *(cause H S)*) can be learned with confidence (Step 4c). This is the actual case in Example 3, since *A = B = E = (low (flow air carburetor))* and *Y = Z = (not (combustion cylinder))*, and the relationship

```
(CAUSE (not (movable butterfly-valve))(not (run engine)))
```

can be learned.

If either direction is unable to link, the system infers that the gaps between the chains can be filled, based on the expert instructor having given the hypothesis and the symptom. This is shown in Figure 10(f) where *A and B* represent the same thing, but nothing in the chain *E to Y* matches anything in the chain *S to Z.* The inference is made that smallest gap should be filled (*(cause Y Z)*). This is the second difference from the general algorithm. In Example 3, if the system did not have the information that

```
(CAUSE (low (mix air gas)) (not (combustion cylinder)))
```

available, then *Y = (low (mix air gas))* and *Z = (not (combustion cylinder))*, and the system would infer (with lower plausibility than other things learned) that the relationship

```
(CAUSE (low (mix air gas)) (not (combustion cylinder))).
```

holds.

Consistency

Detection of Inconsistencies

The system resolves contradictions in the causal domain model by checking for a contradiction whenever causal information is added. The check is only done for the frame where the information is being added; no *causal chaining* is done. For example, if a relationship

```
    (CAUSE (low (mix air gas)) (not (combustion cylinder)))
```

exists in the causal domain model indexed under the carburetor, and a somewhat contradictory relationship

```
    (CAUSE (low (mix air gas)) (low (combustion cylinder)))
```

is added to the causal domain model under the cylinders, the contradiction will not be detected. This is necessary computationally, since contradictions could exist in any number of somewhat related locations. Also, the contradiction could be more subtle, where it would take several steps of *causal chaining* before it is discovered. Section 4.3.2 discusses this more fully. Exhaustive checking would be highly inefficient. This is in accord with the algorithm proposed by Johnson-Laird [1983], in that contradictions are dealt with only when necessary, when parts of a causal domain model are connected. However, it differs in that his process, once contradictions are detected, exhaustively tests the model for consistency.

Handling of Inconsistencies

If a conflict exists, the piece of information with the higher credibility will be retained. If

```
    (CAUSE (low (mix air gas)) (low (combustion cylinder)))
```

is indexed under the carburetor with a fairly high plausibility, and the system attempts to add

```
    (CAUSE (low (mix air gas)) (high (combustion cylinder)))
```

under the carburetor, with a low plausibility, then the existing relationship will be retained and the new possibility discarded. On the other hand, relationships given by the instructor, such as

```
    (CAUSE (low (flow air carburetor))(low (mix air gas)))
```

are given high plausibility and would replace a relationship incorrectly learned by the system such as

```
    (CAUSE (low (flow air carburetor))(high (mix air gas)))
```

if it was indexed under the same component (carburetor in this case).

3.4.3 The Effect of Learning

EDSEL2 shows two effects of learning:

- *In succeeding episodes different information will be learned.*
- Succeeding diagnosis will be better and/or faster.

Succeeding Learning Episodes

The system learns different information when the causal domain model is in a different state. Figures 11, 12, 13, 14, and 15 demonstrate this principle. Figure 11 shows a subset of the causal knowledge in the working model before one learning episode. Figure 12 shows a simplified trace of the system's processing of a learning episode. Figure 13 shows the causal knowledge subset modified based on the learning episode. Figure 14 shows part of the trace of a second run, after the knowledge has been added to the model, and Figure 15 shows the further modified model subset.

The initial state of part of the model is shown in Figure 11. Only the causal relationships are shown, and only a few frames. In the processing shown in Figure 12, the system first processes the symptom

```
    (NOT (CRANK ENGINE-SYSTEM))
```

and builds a chain of related findings through

```
    (NOT (CRANK CRANKSHAFT)),
    (NOT (CRANK STARTER)).
```

to

```
    (NOT (CRANK STARTER-GEAR))
```

(The clause `(NOT (CONNECTED STARTER BATTERY-CABLES))` is a different possible branch in the chaining, which will not be of interest in this case). Next the system processes the instructor's hypothesis (CORRODED BATTERY-TERMINALS). Since the causal domain model did not have any causal knowledge related to battery terminals at that time, the system is not able to build a chain to further findings. The chains do not meet, so an explanation is necessary. The instructor's explanation is that battery terminals are frequently corroded. Once again, the lack of causal knowledge prevents any *causal chaining.* At this point, based on the instructor's status as expert, the system infers that the explanation is relevant to the hypothesis, and the system bridges the gap, learning the plausible relationship

```
    ((CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7)),
```

as illustrated in Figure 10(f). This relationship is stored under both BATTERY-TERMINALS and STARTER-GEAR. Note these relationships in Figure 13.

When the system is re-run using the same learning episode and the modified working model, additional information is learned. In the processing shown in Figure 14, the system is able to build the chain further back from the symptom because of the newly learned causal knowledge under starter gear. It chains from

```
    (NOT (CRANK STARTER-GEAR))
```

to

```
    (CORRODED BATTERY-TERMINALS).
```

Then, because of the new information under battery terminals, the system can chain from the hypothesis. These chains meet, so the hypothesis can be associated with the symptom in the symptom-fault table, and the causal domain model can be updated to include that the hypothesis causes the symptom. In this case, that is:

```
    (CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK ENGINE-SYSTEM))).
```

The explanation then is not necessary, and does not cause anything to be added. The processing of the explanation has been omitted from Figure 14. Figure 15 shows the new relationship added under BATTERY-TERMINALS and ENGINE-SYSTEM.

Succeeding Diagnoses

As a result of learning, not only is the Learning by Understanding Explanations system prepared to learn additional things because more causal knowledge allows further inferencing, but the diagnosis performance element is better able to diagnose. It uses symptom fault sets to index into the causal domain model. Using the portion of the causal domain model that represents the component suggested as a possible problem, it tries to verify the possibility by explaining the link from the symptom to the fault. If it can do so, it then suggests a test of that component. For example, if the symptom fault table suggested the possible association of

```
    (CORRODED BATTERY-TERMINALS)
```

with the symptom of

```
    (NOT (CRANK ENGINE-SYSTEM)),
```

then the diagnostic system would try to verify that that is reasonable by forming an explanatory chain. With the model in the state shown in Figure 11, before any learning, this would not be possible, but with the model in the state shown in Figure 15 it would. Once the possibility has been verified, a test must be carried out to see if that is actually the case. That test will be found in the causal domain model indexed under the appropriate component.

Future Directions

The implementation has some problems. The model is too dependent on explicit causes, therefore too fragile, and not flexible enough. The system needs to know about generalized actions/functions. For example, this will allow the system to make inferences based on what can be a problem with gears or belts or wires in general. The current model allows little generalization. Also, the model needs to better handle enabling states for actions. For example, a valve being open will under normal circumstances cause the amount of a substance outside to decrease and the amount of a substance inside to increase. However, before that inference is made, there must be some amount of substance outside the valve. Forbus's Qualitative Process Theory [1984] handles some things like this nicely. The causal domain model may also need different tracks in order to handle different types of cars. The handling of bad data in the causal domain model is crude, as is the handling of branching *causal chains*. It also seems that the student following along should make use of his symptom fault knowledge. Lastly, the plausible inference made when chains do not meet should be restricted based on some general knowledge as in the other implementation, or based on some relatedness heuristics as in Doyle [1984], Pazzani [1987] and Hammond [1988].

The diagnosis program was not the main focus of this effort, more thought will need to be given to it.

3.4.4 Summary of the Implementations

Both EDSEL1 and EDSEL2 started with the same problem: how to learn from explanations and hypotheses given in response to a symptom. Both used the same input examples in roughly similar formats. They differed in the way they achieved some of the implementation goals. The biggest difference in the implementations was in the representations. EDSEL1 used a fine grained representation similar to local connectionist models and KODIAK [Wilensky, 1986], while EDSEL2 used a frame-based representation.

It is surprising that the two implementations, which began as vastly different representations of the same problem, have started to converge at many levels. As the implementations proceeded, the fine grained approach has come to use the conventions of frame-based systems, and the knowledge-based approach has come to use a more homogeneous representation for all system knowledge. Similarly, through independent development, both implementations have come to need some representation of the credibility of causal knowledge. EDSEL1 uses weights, while EDSEL2 uses certainty factors. However, these have become roughly equivalent.

Of more practical importance for future research, however, are the differences. There are two general differences in the technical details of how the general algorithm was implemented. First, when a *causal gap* is present but no explanation fills that gap, EDSEL1 uses generic knowledge about what affects what, whereas EDSEL2 uses a less general but far simpler notion of filling gaps between recent hypotheses and symptoms. The first method brings more knowledge to bear on the postulation of a new causal relationship. It verifies that the proposed relationship is not inconsistent with any known knowledge, and is potentially relevant. It is therefore a more flexible and reliable metric for evaluating whether a given *causal gap* should be filled.

The approaches are somewhat different in how the *causal chains* are stored. EDSEL2 keeps flat contexts which contain lists of what information has been encountered in the current *causal chaining*. For example, if the system is processing a hypothesis, the context for that hypothesis will be updated as causal information is used to consider the implications of the hypothesized fault. Any implications are added to the list. EDSEL1 keeps recent chains of reasoning. The tree form of any explorations is retained for future reasoning with a given example. Thus, if a subsequent hypothesis requires similar reasoning, that reasoning is available. A disadvantage of this approach is that if there are many plausible hypotheses at a given time, the program will pursue only a few to great depth.

The third difference involves where causal information is stored and how it can be accessed. EDSEL1 does not address the issue of limited availability of causal relationships. Instead, if a relationship is in the system, it can be used. On the other hand, EDSEL2 allows the more realistic situation in which causes are not maximally indexed when they enter the system. That is, they may not necessarily be retrieved when needed unless the proper cues are present. This is a more realistic and efficient approach for a system with a very large memory.

3.5 Related Work

As has already been noted, the current effort uses techniques that are in some ways similar to those of Dejong and Mooney [1986] and Mitchell et al. [1986]. It is similar in that it recognizes that a system may know all the necessary information to solve a given problem, but a solution path through this information may be inefficient to calculate. Instruction or observation might allow a sample path to be generated and condensed to a more easily used form. Specifically, in diagnosis, a *causal chain* must be discovered in a potentially very large network of causal information. EBL can be profitably used to permit instruction to produce "short cuts" in that network.

Classical EBL, however, does not produce enough learning when the causal network is incomplete. Hall's [1986,1988] learning by failing to explain (LBFE) approach was an attempt to remedy this problem. The technique involves isolating the information that is present in the input but that is not understood, and subsequently adding it to the existing EBL system. Our approach differs from Hall's work by proposing that the information that must be added in the absence of an explanation is not necessarily explicitly represented in the input. Instead, the current effort allows for more general assignment of blame by disambiguating what might be implied by the instructor. Also, the current effort presents a domain independent notion of learning by attempting to explain that describes how potentially any diagnostic causal model might grow, whereas Hall's effort was, in his own view, domain specific.

One of the methods that is used to augment incomplete networks in the current approach is to use relationships that are more general than causes in order to infer causal relations. For example, an action and a state change that relate to the same object tend to be causally related. This technique was originally used by Pazzani [1987] and similar approaches were suggested by Russell [1987], Doyle [1984], and Hammond [1988].

Rajamoney and DeJong [1987] have specifically addressed the problem of inconsistencies or missing information in a causal model for simulation. If more than one simulation is possible, their system will experimentally search for disambiguating features in the environment. Although this approach is clearly useful, it does not allow for modification of the general causal information in the model. It concentrates on quantitative values for the current situation, and does not learn any general knowledge.

Another related work is Mitchell et al's [1985] Learning Apprentice (LEAP) system. LBUE and LEAP are similar in that they both use explanations to properly modify knowledge when given instruction. They differ on how the explanation is related to the knowledge that is learned. LEAP uses two distinct types of knowledge – domain knowledge and implementation knowledge – whereas, LBUE has a single type that is used both for explanation and for performance. LEAP uses its domain knowledge to learn new implementation rules. LBUE allows the system to improve its explanation capability because what is learned can be used in future explanations. A related difference is that LBUE handles incomplete/imperfect domain theories, while LEAP assumes a complete theory. Their domain is less complex since training examples can be proved correct or incorrect, unlike examples in medical or diagnostic domains.

Work on Qualitative Models by Forbus [1984], DeKleer and Brown [1981], and Kuipers [1984], shares the goal of representing a physical domain qualitatively in order to reason about it. However, they do not attempt to have the system augment its model. In addition, with the exception of the medical modeling by Kuipers, the domains are significantly less complex than a car. Chandrasekaran and Mittal [1982], used a causal model of a medical domain to generate compiled rules or associations. However, just as in EBL, they do not add any truly new knowledge.

Other studies that have shared some of the goals of the current research in other domains include, Haas and hendrix [1983, *Learning* by being told], and Sammut and Banerji [1986, Concept learning by asking questions].

3.6 Conclusions and Future Directions

This paper has discussed the Learning by Understanding Explanations paradigm that was observed by Lancaster and Kolodner [1987, 1988] in the training of car mechanics. The main contribution of LBUE is the ability to accept new knowledge into the causal domain model and to create new generalizations by understanding an instructor's example problem. In particular, an algorithm for learning to diagnose is proposed which when provided with an incomplete causal domain model, symptom-fault pairings, and an expert's diagnostic example, will output a more complete model. The algorithm is capable of noticing missing causal relations, general relations, objects, and efficiency information. Once noticed, this information can be added to the model. The algorithm is further capable of correcting inconsistent knowledge. This algorithm has been implemented in two programs, EDSEL1 and EDSEL2, that demonstrate the concept and help to explore its possibilities.

There are several directions for future research. First, better representation of the causal knowledge is needed to take full advantage of the inferencing possible from qualitative models. One aim is to use more levels of abstraction to allow reasoning at whatever level may be appropriate, and so that knowledge of the normal function of a mechanism can be more useful in inferencing. For example, in some cases it may be appropriate to reason at the level of the starter, at other times, at the level of the starter motor, and at yet other times, at the level of the starter brushes. A related proposal is the representation of normal function on a more detailed level, rather than at a single level of abstraction. Currently, components are represented with inputs, outputs, connectedness, parts, part-of, and causal relationships. It would be more elegant if the normal function was represented as a general function that was common between several components, that would have common effects, common enabling states, common obstacles to their function and so on. Then concepts that are learned can be better generalized. For example, if the system learned that the starter pinion gear has worn teeth which impairs its function, then the generalization that worn teeth is an obstacle to the functioning of a gear can be learned. A further improvement in the model would involve better handling of conditional causal relationships. For example, a valve being open normally causes more of a substance to be at the destination. However, this only occurs if there is some of the substance at the source, and the destination is not already full.

Second, the EDSEL algorithm should include understanding of why particular tests were done by the instructor and what the results of the tests mean for previous and future hypotheses. Tests are frequently linked both to a preceding hypothesis that led to its execution, and a following hypothesis that is a reaction to its result. We want to investigate how the process of diagnosis can be learned, not just how a system can improve its answer or result. A long term goal is to use this knowledge in intelligent tutoring.

Third, it may be possible to incorporate Case-Based Reasoning [Kolodner and Simpson, 1984], which is a method of using previous episodes and evaluation of their results to suggest solutions to new problems. One general method that seems promising has been suggested by Koton [1988a, 1988b]. Associational knowledge is used first, then if necessary, previous cases are used. Finally, if a diagnosis has not been made, the model can be used to make the diagnosis. This has the advantage of using the fastest method that will be successful in any given case. The same learning episodes could create associations through collapsing chains and/or forming generalized episodes, while exceptional cases would be stored and the causal model be augmented in an LBUE like manner. We would want to differ from Koton's approach in that she has all the test results available at the start, and as mentioned above we want to investigate the process of diagnosis as well as the result. This bears further investigation. Another general method would be to store cases based on correlated attributes in which a single domain model could be used for LBUE and for case storage. The greatest potential advantage of this approach is that LBUE and case use would be naturally balanced. Correlated attribute methods involve storing cases in an overlapped format for maximal generalization at the cost of perfect case recall. Other approaches that may be worth examining are the use by Hammond [1988] and Barletta and Mark [1988] of EBL type methods for improving regular Case-based reasoning methods.

Fourth, in diagnosis, *causal chaining* is not the only strategy used, though it was the most common in the protocols. The explanations used by the instructor reflect several different strategies. For example, a dead battery could be hypothesized as a problem because that occurs frequently. It may be that a particular model of car has a defect. Or the explanation may be an explanation of a normal function of a component that the instructor thinks the student may not understand. For this reason, and to allow strategies to be learned and improved, diagnostic strategies must be explicitly represented. Some initial work has been done on the representation. This is of course related to understanding the relationship between tests and hypotheses as well.

```
Generalized Symptom-Fault Sets:

 Symptom: ((NOT (START)) CAR)
  Faults: ((MALFUNCTION FUEL-SYSTEM), credibility = 3)
          ((MALFUNCTION STARTING-SYSTEM), credibility = 1)
          ((MALFUNCTION AIR-INTAKE-SYSTEM), credibility = 1)

 Symptom: (CRANK ENGINE-SYSTEM SLOW)
  Faults: ((MALFUNCTION STARTING-SYSTEM), credibility = 4)
          ((MALFUNCTION CHARGING-SYSTEM), credibility = 2)

 Symptom: (MALFUNCTION FUEL-SYSTEM):
  Faults: ((MALFUNCTION FUEL-LINE), credibility = 4)
          ((MALFUNCTION FUEL-TANK), credibility = 2)
          ((MALFUNCTION FUEL-PUMP), creaibility = 1)
```

Figure 1: Generalized Symptom/Fault Sets. 'Credibility' represents the likelihood that a particular pair holds.

Figure 2 - Representation of Protocol

```
(defvar *protocol#13*) ; Made up protocol
(setq *protocol#13*
        '((SYM (Not (Crank Engine-system)))
          (HYP H1 (Not (Contains Fuel-Tank Gasoline)))
          (TEST (Contains Fuel-Tank Gasoline) (Type Visual)
                (Result (Not (Contains Fuel-Tank Gasoline))))
          (HYP H2 (Corroded Battery-Terminals))
          (EXP H2 (Frequency (Corroded Battery-Terminals) High) )
          (EXP H2 (Cause (Corroded Battery-Terminals)
                         (Not (Connected Battery Electrical-wire))))
          (TEST (Corroded Battery-Terminals) (Type Visual)
                (Result (Corroded Battery-Terminals)))
          (ACTION (Scrape Battery-Terminals Corrosion))
          (TEST (Corroded Battery-Terminals) (Type Visual)
                (Result (Not (Corroded Battery-Terminals))))
          (HYP H3 (Loose (Connected Battery-Cables Ground)))
          (TEST (Connected Battery-Cables Ground) (Type Visual)
                (Result (Loose (Connected Battery-Cables Ground))))
          (HYP H4 (Loose (Connected Battery-Cables Solenoid)))
          (EXP H4 (Cause (Loose (Connected Battery-Cables Solenoid))
                         (Not (Adjacent Starter-Gear Flywheel))))
          (TEST (Connected Battery-Cables Solenoid) (Type Visual)
                (Result (Loose (Connected Battery-Cables Solenoid))))
          (HYP H5 (Low (Power Battery)))
          (TEST (Power Battery) (Type VAT40) (Result (Low (Power Battery))))
          (HYP H6 (Not (Generate Battery-Cell Power)))
          (EXP H6 (Cause (Spill Battery-Cell Electrolytes)
                         (Not (Generate Battery-Cell Power))))
          (FAULT (Not (Generate Battery-Cell Power)))
        ))
```

```
Starter:
       (isa component)
            ;A starter is a component.
       (part-of starting-system)
            ;A starter is a part of the starting system.
       (input electricity battery battery-cables)
            ;A starter receives electricity from
            ;the battery via battery cables.
       (parts starter-pinion-gear starter-motor)
            ;A starter has parts: pinion gear and
            ;starter motor.
       (function spin-action starter-pinion-gear)
            ;The function of the starter is to
            ;spin the pinion gear.
       (cause (switch-action solenoid on)
              (adjacent starter-pinion-gear
                              flywheel-ring-gear))
            ;Setting the solenoid switch causes
            ;the pinion gear and the flywheel gear
            ;to become adjacent.
       (cause (crank starter-pinion-gear)
              (crank flywheel-ring-gear))
            ;Cranking the pinion gear causes the
            ;flywheel gear to crank.
```

Figure 3: A Generalized causal domain model Definition of a Starter.

```
Carburetor-barrel:
       (isa ^conduit)
       (parts ^venturi ^carburetor-barrel-wide-part1
              ^carburetor-barrel-wide-part2)
       (part-of ^carburetor))
       (input ((^fuel ^carburetor-float-bowl ^carburetor-pipe-to-venturi)
               (^air ^vacuum-line)))
       (output ((^fuel-air-mix ^intake-manifold)))
       (connected-to
               ((^carburetor-float-bowl ^carburetor-pipe-to-venturi)
                (^vacuum-line)
                (^intake-manifold)))
       (function
               ((^mix-action ^air ^fuel)
                (^conduit-action ^air ^vacuum-line ^intake-manifold)))
       (cause (contains ^vacuum-line ^air)
              (increase (contains ^intake-manifold ^air)))
       (cause (clogged ^carburetor-barrel)
              (not (increase (contains ^intake-manifold ^air))))
       (cause (hole ^carburetor-barrel)
              (not (increase (contains ^intake-manifold ^air))))
       (cause (low (contains ^carburetor-barrel air-pressure))
              (increase (contains ^carburetor-barrel ^fuel))
       (cause (high (contains ^carburetor-barrel air-pressure))
              (lean ^fuel-air-mix))

Carburetor-barrel-wide-part1:
       (isa ^conduit)
       (parts )
       (part-of ^carburetor-barrel)
       (input ((^air ^vacuum-line)))
       (output ((^air ^venturi)))
       (connected-to
               ((^venturi)
                (^vacuum-line)))
       (function
               ((^conduit-action ^air ^vacuum-line ^venturi)))
       (cause (contains ^vacuum-line ^air)
              (increase (contains ^venturi ^air)))
       (cause (clogged ^carburetor-barrel-wide-part1)
              (not (increase (contains ^venturi ^air))))
       (cause (hole ^carburetor-barrel-wide-part1)
              (not (increase (contains ^venturi ^air))))

Carburetor-barrel-wide-part2:
       (isa ^conduit)
```

```
        (parts )
        (part-of ^carburetor-barrel)
        (input ((^fuel-air-mix ^venturi)))
        (output ((^fuel-air-mix ^intake-manifold)))
        (connected-to
                ((^venturi)
                 (^intake-manifold)))
        (function
                ((^conduit-action ^fuel-air-mix ^venturi ^intake-manifold)))
        (cause (contains ^venturi ^fuel-air-mix)
               (increase (contains ^intake-manifold ^fuel-air-mix)))
        (cause (clogged ^carburetor-barrel-wide-part2)
               (not (increase (contains ^intake-manifold ^fuel-air-mix))))
        (cause (hole ^carburetor-barrel-wide-part2)
               (not (increase (contains ^intake-manifold ^fuel-air-mix))))

Venturi:
        (isa ^conduit)
        (parts )
        (part-of ^carburetor-barrel)
        (input ((^fuel ^carburetor-float-bowl ^carburetor-pipe-to-venturi)
                (^fuel ^accelerator-pump ^carburetor-discharge-check-valve)
                (^air ^carburetor-barrel-wide-part1)))
        (output
               ((^fuel-air-mix ^carburetor-barrel-wide-part2)))
        (connected-to
               ((^carburetor-pipe-to-venturi)
                (^carburetor-discharge-check-valve)
                (^carburetor-barrel-wide-part1)
                (^carburetor-barrel-wide-part2)))
        (function
               ((^mix-action ^air ^fuel)
                (^conduit-action ^air ^carburetor-barrel-wide-part1
                                      ^carburetor-barrel-wide-part2))
                (^pressur -flow-action ^fuel ^carburetor-float-bowl ^venturi)))
        (cause (contains ^carburetor-barrel-wide-part1 ^air)
               (increase
                  (contains ^carburetor-barrel-wide-part2 ^fuel-air-mix)))
        (cause (clogged ^venturi)
               (not (increase
                        (contains ^carburetor-barrel-wide-part2 ^fuel-air-mix))))
        (cause (hole ^venturi)
               (not (increase
                        (contains ^carburetor-barrel-wide-part2 ^fuel-air-mix))))
        (cause (low (contains ^venturi air-pressure))
               (increase (contains ^venturi ^fuel))
        (cause (high (contains ^venturi air-pressure))
               (lean ^fuel-air-mix))
```

Figure 4: EDSEL2's causal domain model Definition of a Carburetor barrel.

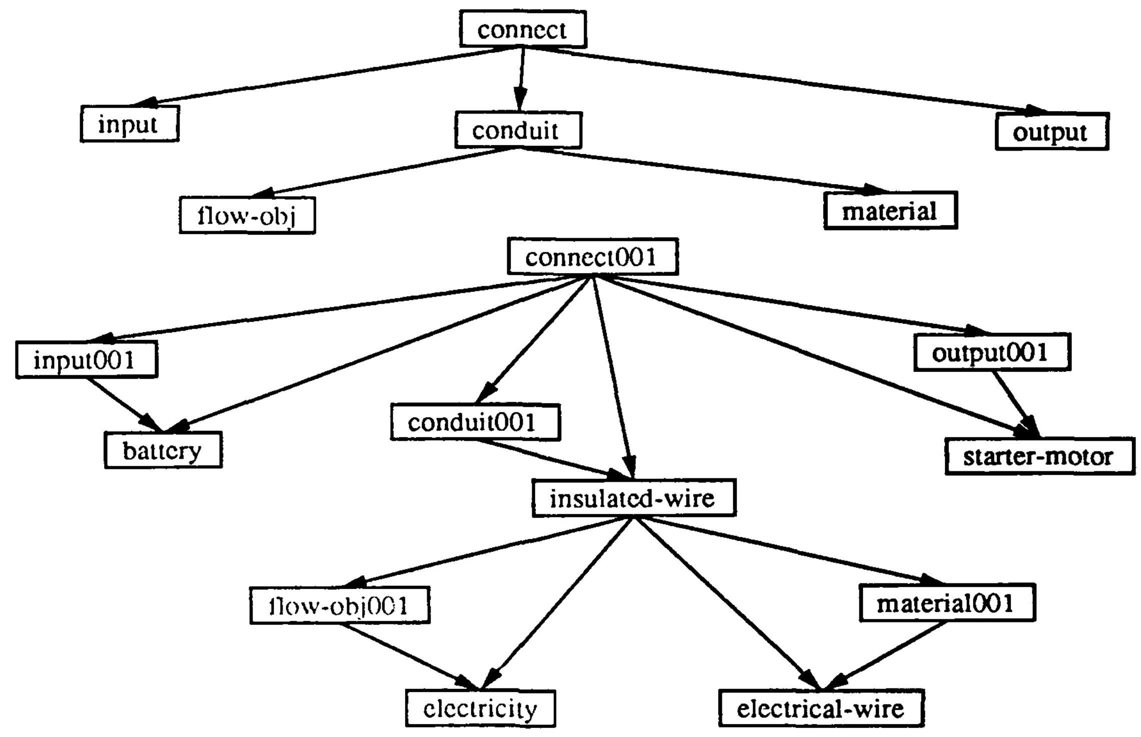

Figure 5: Representation of frame-like information for the first implementation.

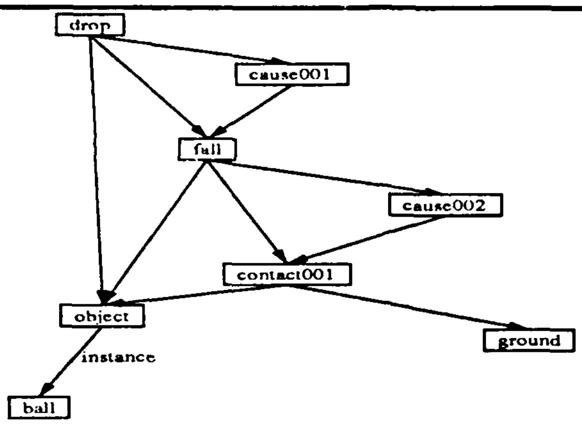

Figure 6: Causal representations for EDSEL1.

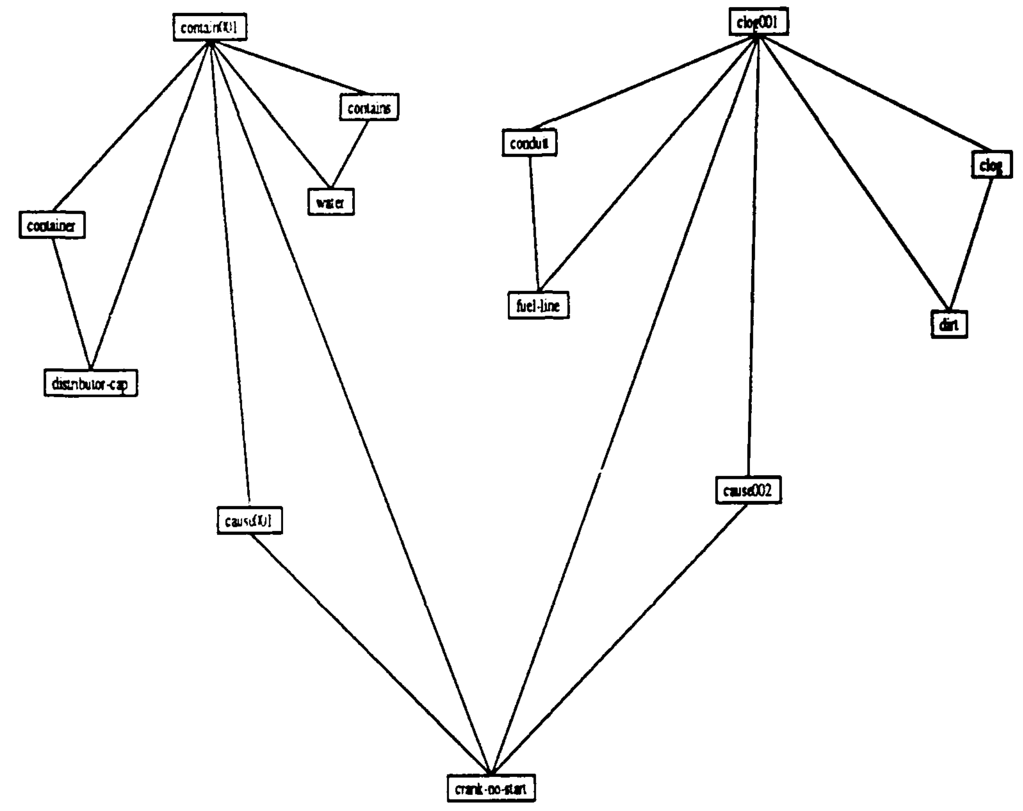

Figure 7: An example of causal competition.

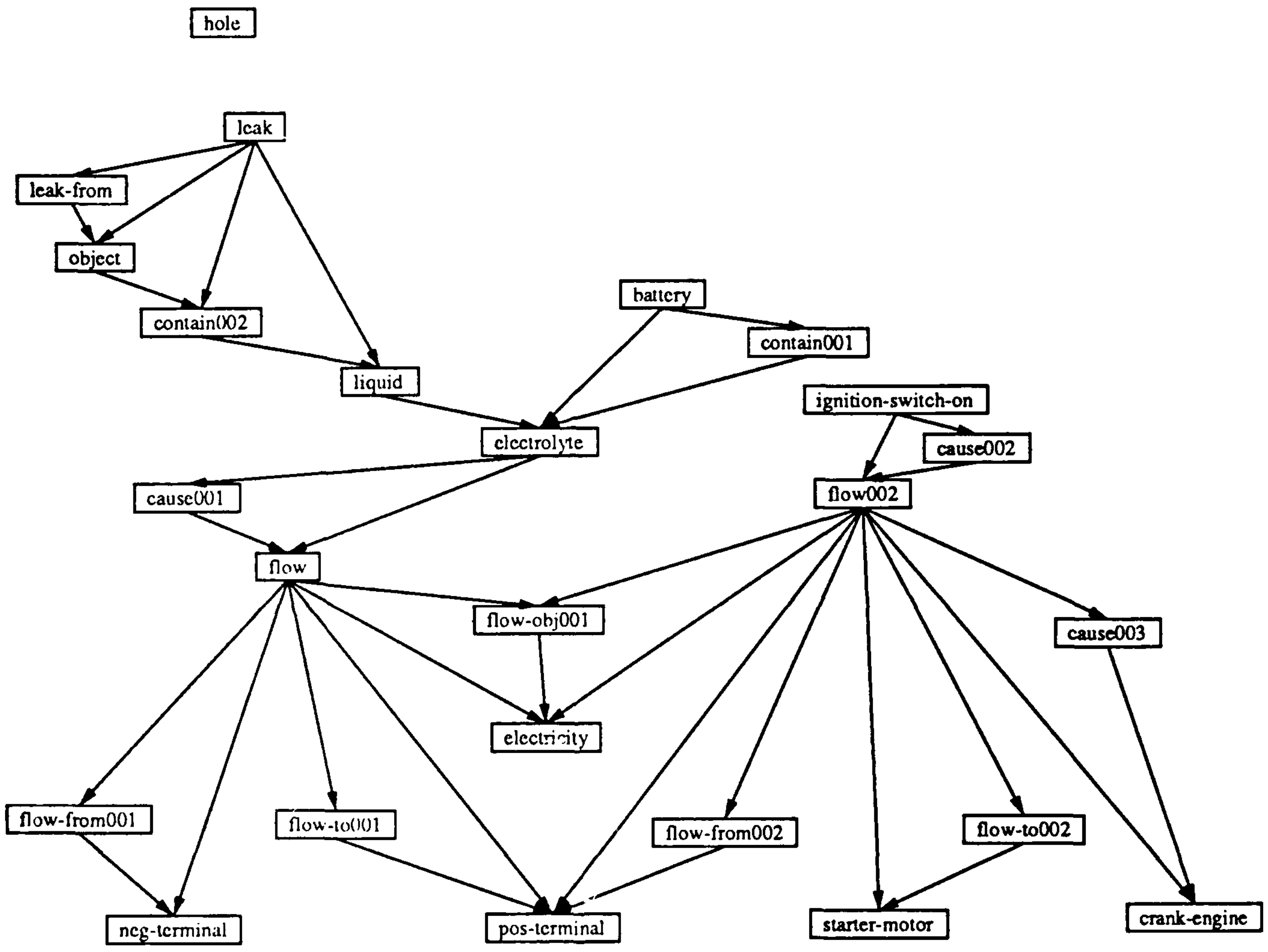

Figure 8: Partial state of the causal domain model before instruction.

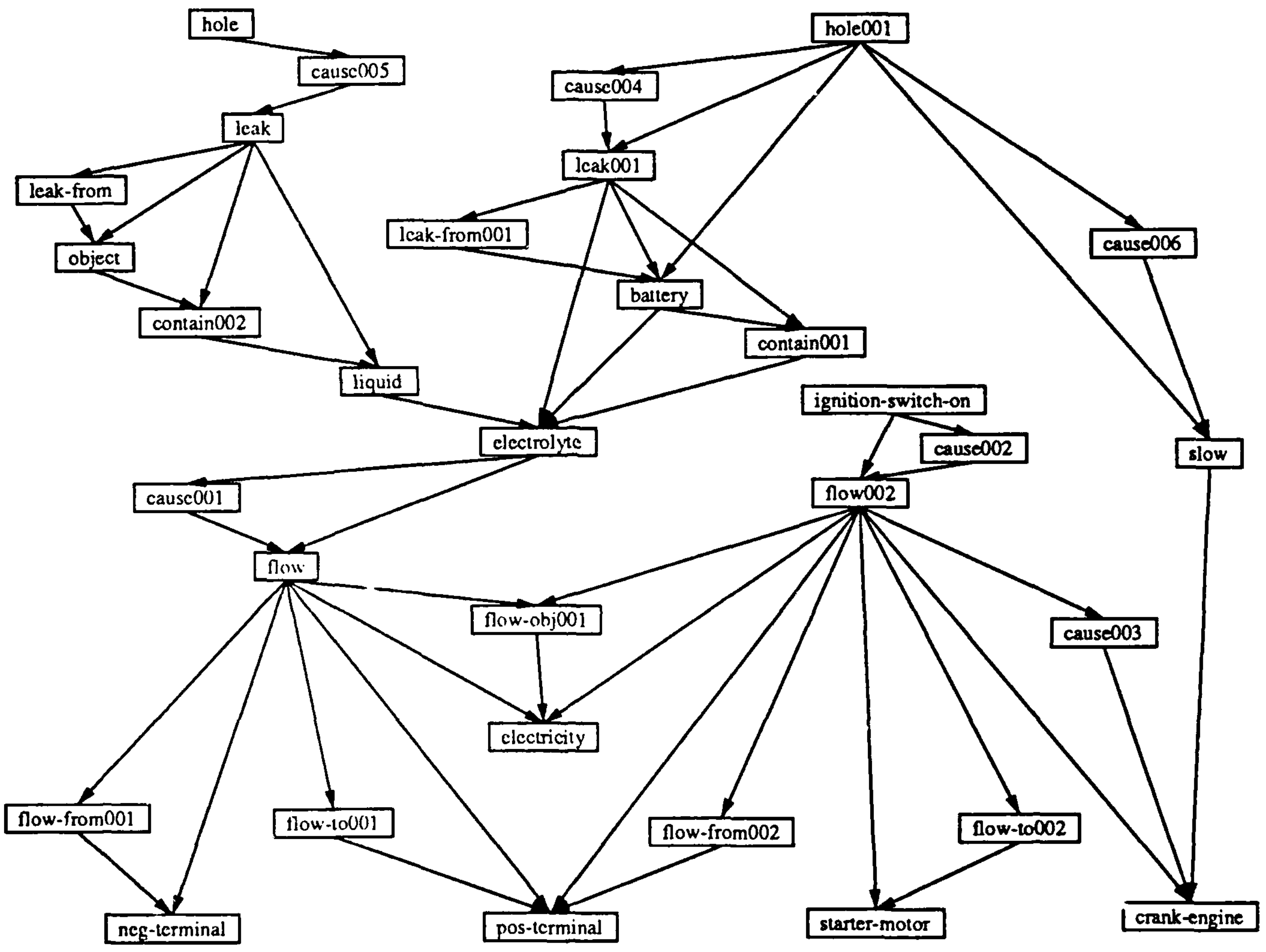

Figure 9: Partial state of the causal domain model after instruction.

Figure 10 - EDSEL2 *Causal Gap* Filling

The uppercase letters represent pieces of knowledge that are involved in causal chains.

H - Hypothesis
S - Symptom
E - Explanation
A - Edge of chain forward from hypothesis
B - Edge of chain back from explanation
Z - Edge of chain back from symptom
Y - Edge of chain forward from explanation

```
                      H       AZ    S
                      .-------><----.
   Since A=Z, Learn:   (cause H S)
                             (a)

                    H       A   Z    S
                    .------->   <----.
                             (b)

                 H       A   B  E   Y   Z    S
                 .------->   <--.--->   <----.
                             (c)

                 H       AB  E   Y   Z    S
                 .-------><--.--->   <----.
   Since A=B, Learn:   (cause H E)
                             (d)

                 H       AB  E   YZ    S
                 .-------><--.---><----.
   Since A=B and Y=Z, Learn:  (cause H S)
                             (e)

                 H       AB  E   Y   Z    S
                 .-------><--.--->   <----.
   Since Y^=Z, Learn:   (cause Y Z)
                             (f)
```

Figure 11 - Some causal domain model Relationships Before Learning

Causal relationships only; with credibilities.

EQUIV means that the two clauses are equivalent;

CAUSE represents a straight causal relationship;

NEC-CAUSE represents a necessary cause;

SUFF-CAUSE represents a sufficient cause;

POSS-CORR represents a positive correlation between clauses.

```
********************
      ENGINE-SYSTEM
********************
(EQUIV (CRANK ENGINE-SYSTEM) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK STARTER) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (TURN DRIVESHAFT) 9)
********************

********************
      CRANKSHAFT
********************
(POSS-CORR (CRANK CRANKSHAFT) (AMOUNT COMBUSTION) 9)
(CAUSE (CRANK STARTER) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (MOVEMENT PISTONS) 9)
(CAUSE (CRANK CRANKSHAFT) (MOVEMENT CAMSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (TURN DRIVESHAFT) 9)
********************

********************
      STARTER
********************
(CAUSE (SWITCH-ACTION SOLENOID ON) (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR) 6)
(NEC-CAUSE (CRANK STARTER-GEAR) (CRANK CRANKSHAFT-GEAR) 8)
(NEC-CAUSE (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR) (CRANK CRANKSHAFT-GEAR) 8)
(NEC-CAUSE (CONNECTED STARTER BATTERY-CABLES) (CRANK STARTER) 9)
(SUFF-CAUSE (AND (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR)
                 (CRANK STARTER-GEAR))
            (CRANK CRANKSHAFT-GEAR) 8)
(POSS-CORR (AMOUNT ELECTRICITY) (CRANK STARTER-GEAR) 7)
(EQUIV (CRANK STARTER) (CRANK STARTER-GEAR) 7)
(EQUIV (CRANK CRANKSHAFT-GEAR) (CRANK CRANKSHAFT) 7)
********************

**********************
      STARTER-GEAR
**********************
**********************

**********************
      BATTERY-TERMINALS
**********************
**********************
```

Figure 12 - Learning By Understanding Explanations Processing

```
Processing Protocol #13a of instructor. Protocol is:

(SYM (NOT (CRANK ENGINE-SYSTEM)))
(HYP H2 (CORRODED BATTERY-TERMINALS))
(EXP H2 (FREQUENCY (CORRODED BATTERY-TERMINALS) HIGH))

*****************************************************************************
Processing symptom  (NOT (CRANK ENGINE-SYSTEM))

Extending symptom chain from:  (NOT (CRANK ENGINE-SYSTEM)) 10
Extending symptom chain from:  (NOT (CRANK CRANKSHAFT)) 9
Extending symptom chain from:  (NOT (CRANK STARTER)) 8
Extending symptom chain from:  (NOT (CONNECTED STARTER BATTERY-CABLES)) 7
Extending symptom chain from:  (NOT (CRANK STARTER-GEAR)) 7

*****************  END PROCESSING  (SYM (NOT (CRANK ENGINE-SYSTEM)))  ********
*****************************************************************************
Examining hypothesis   (H2 (CORRODED BATTERY-TERMINALS)) hyp-list24

Extending hypothesis chain from:  (CORRODED BATTERY-TERMINALS) 9
Hypothesis and Symptom chains do not meet

*****************  END PROCESSING  (HYP H2 (CORRODED BATTERY-TERMINALS))  ******
*****************************************************************************
Processing Explanation  (H2 (FREQUENCY (CORRODED BATTERY-TERMINALS) HIGH))

Try to link (CORRODED BATTERY-TERMINALS) to something in current hypothesis list
And try to link (CORRODED BATTERY-TERMINALS) to something in current symptom list

hypothesis list is:  hyp-list24
     it contains:  (((CORRODED BATTERY-TERMINALS) 9))

symptom list contains:  (((Not (CRANK ENGINE-SYSTEM)) 10) ((Not (CRANK CRANKSHAFT)) 9) ((Not (CRANK STARTER))
8) ((Not (CONNECTED STARTER BATTERY-CABLES)) 7) ((Not (CRANK STARTER-GEAR)) 7))

chaining back from  (CORRODED BATTERY-TERMINALS)  toward hypotheses list

Going back - at:  ((CORRODED BATTERY-TERMINALS) 10)

Resulting backwards chain is:  (((CORRODED BATTERY-TERMINALS) 10))
Chain backwards toward hypothesis and chain forward from hypothesis  meet at  (CORRODED BATTERY-TERMINALS)

chaining forward from  (CORRODED BATTERY-TERMINALS)  toward symptom chain

Going forward - at:  (CORRODED BATTERY-TERMINALS) 10

Resulting forwards chain is:  (((CORRODED BATTERY-TERMINALS) 10))
Chain forward toward symptom and chain backward from symptom  do not meet

unable to link BOTH  (CORRODED BATTERY-TERMINALS)  to something in  hypotheses chain, and  (CORRODED BATTERY-TERMINALS)
to something in symptom chain.

so link edge of symptom chain  (((Not (CRANK ENGINE-SYSTEM)) 10) ((Not (CRANK CRANKSHAFT)) 9) ((Not (CRANK
STARTER)) 8) ((Not (CONNECTED STARTER BATTERY-CABLES)) 7) ((Not (CRANK STARTER-GEAR)) 7))  to edge of  chain
forward from explanation to symptom  (((CORRODED BATTERY-TERMINALS) 10))

Adding to memory under  STARTER-GEAR and BATTERY-TERMINALS
((CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7))

**********  END PROCESSING  (EXP H2 (FREQUENCY (CORRODED BATTERY-TERMINALS) HIGH)) ***
```

Figure 13 - causal domain model Relationships After Learning

Causal relationships only; with credibilities.

EQUIV means that the two clauses are equivalent;

CAUSE represents a straight causal relationship;

NEC-CAUSE represents a necessary cause;

SUFF-CAUSE represents a sufficient cause;

POSS-CORR represents a positive correlation between clauses.

```
********************
      ENGINE-SYSTEM
********************
(EQUIV (CRANK ENGINE-SYSTEM) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK STARTER) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (TURN DRIVESHAFT) 9)
********************

********************
      CRANKSHAFT
********************
(POSS-CORR (CRANK CRANKSHAFT) (AMOUNT COMBUSTION) 9)
(CAUSE (CRANK STARTER) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (MOVEMENT PISTONS) 9)
(CAUSE (CRANK CRANKSHAFT) (MOVEMENT CAMSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (TURN DRIVESHAFT) 9)
********************

********************
      STARTER
********************
(CAUSE (SWITCH-ACTION SOLENOID ON) (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR) 6)
(NEC-CAUSE (CRANK STARTER-GEAR) (CRANK CRANKSHAFT-GEAR) 8)
(NEC-CAUSE (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR) (CRANK CRANKSHAFT-GEAR) 8)
(NEC-CAUSE (CONNECTED STARTER BATTERY-CABLES) (CRANK STARTER) 9)
(SUFF-CAUSE (AND (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR)
                 (CRANK STARTER-GEAR))
            (CRANK CRANKSHAFT-GEAR) 8)
(POSS-CORR (AMOUNT ELECTRICITY) (CRANK STARTER-GEAR) 7)
(EQUIV (CRANK STARTER) (CRANK STARTER-GEAR) 7)
(EQUIV (CRANK CRANKSHAFT-GEAR) (CRANK CRANKSHAFT) 7)
********************

********************
      STARTER-GEAR
********************
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7)
********************

********************
      BATTERY-TERMINALS
********************
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7)
********************

Learned:

   STARTER-GEAR
   ------------
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7)

   BATTERY-TERMINALS
   -----------------
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7)
```

Figure 14 - Second run of LBUE, after Learning

```
 Processing Protocol #13a of instructor. Protocol is:

 (SYM (NOT (CRANK ENGINE-SYSTEM)))
 (HYP H2 (CORRODED BATTERY-TERMINALS))
 (EXP H2 (FREQUENCY (CORRODED BATTERY-TERMINALS) HIGH))

*****************************************************************************************
 Processing symptom  (NOT (CRANK ENGINE-SYSTEM))

 Extending symptom chain from:  (NOT (CRANK ENGINE-SYSTEM)) 10
 Extending symptom chain from:  (NOT (CRANK CRANKSHAFT)) 9
 Extending symptom chain from:  (NOT (CRANK STARTER)) 8
 Extending symptom chain from:  (NOT (CONNECTED STARTER BATTERY-CABLES)) 7
 Extending symptom chain from:  (NOT (CRANK STARTER-GEAR)) 7
 Extending symptom chain from:  (CORRODED BATTERY-TERMINALS) 6
***********  END PROCESSING  (SYM (NOT (CRANK ENGINE-SYSTEM)))  ******************
*****************************************************************************************
 Examining hypothesis   (H2 (CORRODED BATTERY-TERMINALS)) hyp-list27

 Extending hypothesis chain from:  (CORRODED BATTERY-TERMINALS) 9
 Extending hypothesis chain from:  (NOT (CRANK STARTER-GEAR)) 8

 Chains meet at  (CORRODED BATTERY-TERMINALS)

 Adding to memory under  BATTERY-TERMINALS and ENGINE-SYSTEM
((CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK ENGINE-SYSTEM)) 3))

 Adding to Symptom-fault table  ((Not (CRANK ENGINE-SYSTEM)) (((CORRODED BATTERY-TERMINALS)
(3 3))))

********** END PROCESSING  (HYP H2 (CORRODED BATTERY-TERMINALS))  ****************
*****************************************************************************************
 Processing Explanation  (H2 (FREQUENCY (CORRODED BATTERY-TERMINALS) HIGH))

 Try to link  (CORRODED BATTERY-TERMINALS)  to something in current hypothesis list
 And try to link  (CORRODED BATTERY-TERMINALS)  to something in current symptom list
                          .
                          .
                          .
                          .
```

Figure 15 - Causal Domain Model Relationships After More Learning

Causal relationships only; with credibilities.

EQUIV means that the two clauses are equivalent;

CAUSE represents a straight causal relationship;

NEC-CAUSE represents a necessary cause;

SUFF-CAUSE represents a sufficient cause;

POSS-CORR represents a positive correlation between clauses.

```
**********************
      ENGINE-SYSTEM
********************
(EQUIV (CRANK ENGINE-SYSTEM) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK STARTER) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (TURN DRIVESHAFT) 9)
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK ENGINE-SYSTEM)) 3)
********************

**********************
      CRANKSHAFT
********************
(POSS-CORR (CRANK CRANKSHAFT) (AMOUNT COMBUSTION) 9)
(CAUSE (CRANK STARTER) (CRANK CRANKSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (MOVEMENT PISTONS) 9)
(CAUSE (CRANK CRANKSHAFT) (MOVEMENT CAMSHAFT) 9)
(CAUSE (CRANK CRANKSHAFT) (TURN DRIVESHAFT) 9)
********************

********************
      STARTER
********************
(CAUSE (SWITCH-ACTION SOLENOID ON) (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR) 6)
(NEC-CAUSE (CRANK STARTER-GEAR) (CRANK CRANKSHAFT-GEAR) 8)
(NEC-CAUSE (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR) (CRANK CRANKSHAFT-GEAR) 8)
(NEC-CAUSE (CONNECTED STARTER BATTERY-CABLES) (CRANK STARTER) 9)
(SUFF-CAUSE (AND (ADJACENT STARTER-GEAR CRANKSHAFT-GEAR)
                 (CRANK STARTER-GEAR))
            (CRANK CRANKSHAFT-GEAR) 8)
(POSS-CORR (AMOUNT ELECTRICITY) (CRANK STARTER-GEAR) 7)
(EQUIV (CRANK STARTER) (CRANK STARTER-GEAR) 7)
(EQUIV (CRANK CRANKSHAFT-GEAR) (CRANK CRANKSHAFT) 7)
********************

**********************
      STARTER-GEAR
********************
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7)
********************

**********************
      BATTERY-TERMINALS
**********************
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK STARTER-GEAR)) 7)
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK ENGINE-SYSTEM)) 3)
*********************

Learned:

   ENGINE-SYSTEM
   ------------
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK ENGINE-SYSTEM)) 3)

   BATTERY-TERMINALS
   -----------------
(CAUSE (CORRODED BATTERY-TERMINALS) (NOT (CRANK ENGINE-SYSTEM)) 3)
```

4 References

Abbott, K. H. (1988). Robust operative diagnosis as problem solving in a hypothesis space. In *Proceedings of Seventh National Conference on Artificial Intelligence.*

Allison, K. R. (1987). Use of a working model in fault diagnosis. In *Proceedings of the 25th Annual Conference of the Southeast Region ACM.*

Barletta, R. & Mark, W. (1988). Explanation-based indexing of cases. In *Proceedings of a Workshop on Case-Based Reasoning.*

Chandrasekaran, B. & Mittal, S. (1982). Deep versus compiled knowledge approaches to diagnostic problem-solving. In *Proceedings of the National Conference on Artificial Intelligence.*

Collins, A., Salter, W., & Tenney, Y. (198). *Contract progress report.* Technical Report, Bolt, Beranek, and Newman.

DeJong, G. & Mooney, R. (1986). Explanation based learning: an alternative view. *Machine Learning, 1,* 145–176.

DeJong, G. (1983). Acquiring schemata through understanding and generalized plans. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence.*

de Kleer, J. & Brown, J. S. (1981). Mental models of physical mechanisms and their acquisition. In J. R. Anderson (Ed.), *Cognitive Skills and Their Acquisition.* Hillsdale, NJ: Lawrence Erlbaum.

Doyle, R. J. (1984). *Hypothesizing and refining causal models.* Technical Report A.I. Memo No. 811, Massachussetts Institute of Technology.

Fikes, R. E., Hart, P., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence, 3,* 251–288.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence, 24,* 85–168.

Goldstein, I. P. & Bobrow, D. G. (1980). Extending object-oriented programming in smalltalk. In *Proceedings of the 1980 LISP Conference.*

Haas, N. & Hendrix, G. (1983). Learning by being told: acquiring knowledge for information management. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Los Altos, CA: Morgan Kaufmann.

Hall, R. (1986). Learning by failing to explain. In *Proceedings of the National Conference on Artificial Intelligence.*

Hall, R. (1988). Learning by failing to explain: using partial explanations to learn in incomplete or intractable domains. *Machine Learning, 3*, 45–77.

Hammond, K. J. & Hurwitz, N. (1988). Extracting diagnostic features from explanations. In *Proceedings of a Workshop on Case-Based Reasoning.*

Hinrichs, T. R. (1988). Towards an architecture for open world problem solving. In *Proceedings of a Workshop on Case-Based Reasoning.*

Johnson-Laird, P. N. (1983). Mental models. Cambridge, MA: Harvard University Press.

Kolodner, J. & Jr., R. Simpson. (1984). A case for case-based reasoning. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society.*

Koton, P. (1988). Reasoning about evidence in causal explanations. In *Proceedings of a Workshop on Case-Based Reasoning.*

Koton, P. (1988). *Using experience in learning and problem solving.* PhD thesis, Massachussetts Institute of Technology, Cambridge, MA.

Kuipers, J. (1984). Commonsense reasoning about causality: deriving behavior from structure. *Artificial Intelligence, 24*, 169–203.

Lancaster, J. & Kolodner, J. (1987). Problem solving in a natural task as a function of experience. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society.*

Lancaster, J. & Kolodner, J. (1988). Varieties of learning from problem solving experience. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.*

Martin, J. & Redmond, M. (1988). The use of explanations for completing and correcting causal models. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.*

McClelland, J. L. & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: part 1. an account of basic findings. *Psychological Review, 88,* 375–407.

Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The Psychology of Computer Vision.* New York: McGraw-Hill.

Mitchell, T. M., Kellar, R. M., & Kedar-Cabelli, S. T. (1986). Explanation based learning: an unifying view. *Machine Learning, 1,* 47–80.

Mitchell, T. M., Mahadevan, S., & Steinberg, L. I. (1985). Leap: a learning apprentice for vlsi design. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.*

Murphy, G. L. & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review, 92,* 289–316.

Norman, D.A. & Shallice, T. (1986). Attention to action: willed and automatic control of behavior. In R.J. Davidson, G.E. Schwartz, & D. Shapiro (Eds.), *Consciousness and Self-Regulation: Advances in Research Theory (Vol. 4).* New York, NY: Plenum Press.

Pazzani, M. (1987). Inducing causal and social theories: a prerequisite for explanation-based learning. In *Proceedings of the Fourth Annual International Workshop on Machine Learning.*

Rajamoney, S. A. & DeJong, G. F. (1987). *Active ambiguity reduction: An experimental design approach to tractable qualitative reasoning.* Technical Report UILU-ENG-87-2225, University of Illinois at Urbana-Champaign.

Redmond, M. & Martin, J. (1988). Learning by understanding explanations. In *Proceedings of the 26th Annual Conference of the Southeast Region ACM.*

Ross, B.H. (1984). Remindings and their effect in learning a cognitive skill. *Cognitive Psychology, 16,* 371–416.

Russell, S. J. (1987). Analogy and single-instance generalization. In *Proceedings of the Fourth Annual International Workshop on Machine Learning.*

Sammut, C. & Banerji, R. B. (1986). Learning concepts by asking questions. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach, Volume II.* Los Altos, CA: Morgan Kaufmann.

Schank, R.C., Collins, G.C., & Hunter, L.E. (1986). Transcending inductive category formation in learning. *The Behavioral and Brain Sciences, 9,* 639–651.

White, B.Y. & Frederiksen, J.R. (1986). Intelligent tutoring systems based upon qualitative model evolutions. In *Proceedings of the National Conference on Artificial Intelligence.*

Wilensky, R. (1986). *Some problems and proposals for knowledge representation.* Technical Report UCB/CSD 86/294, University of California at Berkeley.